

# Agent Programming with Declarative Goals

F.S. de Boer <sup>ab</sup> K.V. Hindriks <sup>a</sup> W. van der Hoek <sup>c</sup> J.-J.Ch. Meyer <sup>a</sup>

<sup>a</sup> Institute of Information & Computing Sciences, Utrecht  
University, The Netherlands  
P.O. Box 80 089, 3508 TB Utrecht {koenh,frankb,jj}@cs.uu.nl  
<sup>b</sup> National Research Institute for Mathematics and Computer  
Science (CWI), Amsterdam, The Netherlands  
<sup>c</sup> Department of Computer Science, University of Liverpool,  
United Kingdom {wiebe}@csc.liv.ac.uk

## Abstract

A long and lasting problem in agent research has been to close the gap between agent logics and agent programming frameworks. The main reason for this problem of establishing a link between agent logics and agent programming frameworks is identified and explained by the fact that agent programming frameworks have not incorporated the concept of a *declarative goal*. Instead, such frameworks have focused mainly on plans or *goals-to-do* instead of the end goals to be realised which are also called *goals-to-be*. In this paper, a new programming language called GOAL is introduced which incorporates such declarative goals. The notion of a *commitment strategy* - one of the main theoretical insights due to agent logics, which explains the relation between beliefs and goals - is used to construct a computational semantics for GOAL. Finally, a proof theory for proving properties of GOAL agents is introduced. Thus, we offer a complete theory of agent programming in the sense that our theory provides both for a programming framework and a programming logic for such agents. An example program is proven correct by using this programming logic.

## 1 Goal-Oriented Agent Programming

Agent technology has come more and more into the limelight of computer science. Intelligent agents have not only become one of the central topics of artificial intelligence (nowadays sometimes even defined as “the study of agents”, [27]), but also mainstream computer science, especially software engineering, has taken up agent-oriented programming as a new and exciting paradigm to investigate, while industries experiment with the use of it on a large scale, witness the results reported in conferences like Autonomous Agents (e.g. [18]) and books like e.g. [17].

Although the definition of an agent is subject to controversy, many researchers view it as a software (or hardware) entity that displays some form

of *autonomy*, in the sense that an agent is both *reactive* (responding to its environment) and *pro-active* (taking initiative, independent of a user). Often this aspect of autonomy is translated to agents having a *mental state* comprising (at least) *beliefs* on the environment and *goals* that are to be achieved ([35]).

In the early days of agent research, an attempt was made to make the concept of agents more precise by means of *logical systems*. This effort resulted in a number of - mainly - modal logics for the specification of agents which formally defined notions like *belief*, *goal*, *intention*, etc. associated with agents [25, 19, 5, 6]. The relation of these logics with more practical approaches remains unclear, however, to this day. Several efforts to bridge this gap have been attempted. In particular, a number of *agent programming languages* have been developed to bridge the gap between theory and practice [24, 14]. These languages show a clear family resemblance with one of the first agent programming languages Agent-0 [30, 12], and also with the language ConGolog [10, 13, 16].

These programming languages define agents in terms of their corresponding beliefs, goals, plans and capabilities. Although they define similar notions as in the logical approaches, there is one notable difference. In logical approaches, a goal is a *declarative* concept, (also called a *goal-to-be*), whereas in the cited programming languages goals are defined as sequences of actions or *plans* (or *goals-to-do*). The terminology used differs from case to case. However, whether they are called commitments (Agent-0), intentions (AgentSpeak [24]), or goals (3APL [15]) makes little difference: all these notions are structures built from *actions* and therefore similar in nature to *plans*. With respect to ConGolog, a more traditional computer science perspective is adopted, and the corresponding structures are simply called programs. The PLACA language [32], a successor of AGENT0, also focuses more on extending AGENT0 to a language with complex planning structures (which are not part of the programming language itself!) than on providing a clear theory of declarative goals of agents as part of a programming language and in this respect is similar to AgentSpeak and 3APL. The type of goal included in these languages may also be called a *goal-to-do* and provides for a kind of *procedural* perspective on goals.

In contrast, a *declarative* perspective on goals in agent languages is still missing. Because of this mismatch it has not been possible so far to use modal logics which include both belief and goal modalities for the specification and verification of programs written in such agent languages and it has been impossible to close the gap between agent logics and programming frameworks so far. The value of adding declarative goals to agent programming lies both in the fact that it offers a new abstraction mechanism as well as that agent programs with declarative goals more closely approximate the intuitive concept of an intelligent agent. To fully realise the potential of the notion of an intelligent agent, a declarative notion of a goal, therefore, should also be incorporated into agent programming languages.

In this paper, we introduce the agent programming language GOAL (for Goal-Oriented Agent Language), which takes the declarative concept of a goal seriously and which provides a concrete proposal to bridge the gap between theory and practice. GOAL is inspired in particular by the language UNITY

designed by Chandy and Misra [4], be it that GOAL incorporates complex agent notions. We offer a complete theory of agent programming in the sense that our theory provides both for a programming framework and a programming logic for such agents. In contrast with other attempts [30, 34] to bridge the gap, our programming language and programming logic are related by means of a formal semantics. Only by providing such a formal relation it is possible to make sure that statements proven in the logic concern properties of the agent.

## 2 The Programming Language GOAL

In this section, we introduce the programming language GOAL. As mentioned in the previous section, GOAL is influenced by work in concurrent programming, in particular by the language UNITY ([4]). The basic idea is that a set of actions which execute in parallel constitutes a program. However, whereas UNITY is a language based on assignment to variables, the language GOAL is an agent-oriented programming language that incorporates more complex notions such as belief, goal, and agent capabilities which operate on high-level information instead of simple values.

### 2.1 Mental States

As in most agent programming languages, GOAL agents select actions on the basis of their current mental state. A mental state consists of the beliefs and goals of the agent. However, in contrast to most agent languages, GOAL incorporates a *declarative* notion of a goal that is used by the agent to decide what to do. Both the beliefs and the goals are drawn from one and the same logical language,  $\mathcal{L}$ , with associated consequence relation  $\models_C$ . In this paper,  $\mathcal{L}$  is a propositional language, and one may think about  $\models_C$  as ‘classical consequence’. In general however, the language  $\mathcal{L}$  may also be conceived as an arbitrary *constraint system*, allowing one to combine tokens (predicates over a given universe) using the operator  $\wedge$  (to accumulate pieces of information) and  $\exists_x$  (to hide information) to represent constraints over the universe of discourse (Cf. [28]). In such a setting, one often assumes the presence of a constraint solver that tests  $\Gamma \models_C \varphi$ , i.e., whether information  $\Gamma$  entails  $\varphi$ .

Our GOAL-agent thus keeps two databases, respectively called the *belief base* and the *goal base*. The difference between these two databases originates from the different meaning assigned to sentences stored in the belief base and sentences stored in the goal base. To clarify the interaction between beliefs and goals, one of the more important problems that needs to be solved is establishing a meaningful relationship between beliefs and goals. This problem is solved here by imposing a constraint on mental states that is derived from the default commitment strategy that agents use. The notion of a commitment strategy is explained in more detail below. The constraint imposed on mental states requires that an agent does not believe that  $\phi$  is the case if it has a goal to achieve  $\phi$ , and, moreover, requires  $\phi$  to be consistent if  $\phi$  is a goal.

**Definition 2.1** (*mental state*)

A *mental state* of an agent is a pair  $\langle \Sigma, \Gamma \rangle$  where  $\Sigma \subseteq \mathcal{L}$  are the agent's beliefs and  $\Gamma \subseteq \mathcal{L}$  are the agent's goals (both sets may be infinite) and  $\Sigma$  and  $\Gamma$  are such that:

- $\Sigma$  is consistent ( $\Sigma \not\models_C \text{false}$ )
- $\Gamma$  is such that, for any  $\gamma \in \Gamma$ :
  - (i)  $\gamma$  is not entailed by the agent's beliefs ( $\Sigma \not\models_C \gamma$ ),
  - (ii)  $\gamma$  is consistent ( $\Sigma \not\models_C \neg\gamma$ ), and
  - (iii) for any  $\gamma'$ , if  $\models_c \gamma \rightarrow \gamma'$  and  $\gamma'$  satisfies (i) and (ii) above, then  $\gamma' \in \Gamma$

A mental state does *not* contain a program or plan component in the 'classical' sense. Although both the beliefs and the goals of an agent are drawn from the same logical language, as we will see below, the formal meaning of beliefs and goals is very different. This difference in meaning reflects the different features of the beliefs and the goals of an agent. The declarative goals are best thought of as *achievement* goals in this paper. That is, these goals *describe* a goal state that the agent desires to reach. Mainly due to the temporal features of such goals many properties of beliefs fail for goals. For example, the fact that an agent has the goal to *be at home* and the goal to *be at the movies* does not allow the conclusion that this agent also has the conjunctive goal to *be at home and at the movies* at the same time. As a consequence, less stringent consistency requirements are imposed on goals than on beliefs. An agent may have the goal to be at home and the goal to be at the movies simultaneously; assuming these two goals cannot consistently be achieved at the same time does not mean that an agent cannot have adopted both in the language GOAL.

In this paper, we assume that the language  $\mathcal{L}$  used for representing beliefs and goals is a simple *propositional language*. As a consequence, we do not discuss the use of variables nor parameter mechanisms. Our motivation for this assumption is the fact that we want to present our main ideas in their simplest form and do not want to clutter the definitions below with details. Also, more research is needed to extend the programming language with a parameter passing mechanism, and to extend the programming logic for GOAL with first order features.

The language  $\mathcal{L}$  for representing beliefs and goals is extended to a new language  $\mathcal{L}_M$  which enables us to formulate conditions on the mental state of an agent. The language  $\mathcal{L}_M$  consists of so called *mental state formulas*. A mental state formula is a boolean combination of the basic mental state formulas  $B\phi$ , which expresses that  $\phi$  is believed to be the case, and  $G\phi$ , which expresses that  $\phi$  is a goal of the agent.

**Definition 2.2** (*mental state formula*)

The set of *mental state formulas*  $\mathcal{L}_M$  is defined by:

- if  $\phi \in \mathcal{L}$ , then  $B\phi \in \mathcal{L}_M$ ,
- if  $\phi \in \mathcal{L}$ , then  $G\phi \in \mathcal{L}_M$ ,
- if  $\varphi_1, \varphi_2 \in \mathcal{L}_M$ , then  $\neg\varphi_1, \varphi_1 \wedge \varphi_2 \in \mathcal{L}_M$ .

The usual abbreviations for the propositional operators  $\vee$ ,  $\rightarrow$ , and  $\leftrightarrow$  are used. We write **true** as an abbreviation for  $B(p \vee \neg p)$  for some  $p$  and **false** for  $\neg \text{true}$ .

The semantics of belief conditions  $B\phi$ , goal conditions  $G\phi$  and mental state formulas is defined in terms of the classical consequence relation  $\models_C$ .

**Definition 2.3** (*semantics of mental state formulas*)

Let  $\langle \Sigma, \Gamma \rangle$  be a mental state.

- $\langle \Sigma, \Gamma \rangle \models_M B\phi$  iff  $\Sigma \models_C \phi$ ,
- $\langle \Sigma, \Gamma \rangle \models_M G\psi$  iff  $\psi \in \Gamma$ ,
- $\langle \Sigma, \Gamma \rangle \models_M \neg\varphi$  iff  $\langle \Sigma, \Gamma \rangle \not\models_M \varphi$ ,
- $\langle \Sigma, \Gamma \rangle \models_M \varphi_1 \wedge \varphi_2$  iff  $\langle \Sigma, \Gamma \rangle \models_M \varphi_1$  and  $\langle \Sigma, \Gamma \rangle \models_M \varphi_2$ .

We write  $\models_M \varphi$  for the fact that mental state formula  $\varphi$  is true in all mental states  $\langle \Sigma, \Gamma \rangle$ .

A number of properties of the belief and goal modalities and the relation between these operators are listed in Tables 1 and 2. Here,  $\vdash_C$  denotes derivability in classical logic, whereas  $\vdash_M$  refers to derivability in the language of mental state formulas  $\mathcal{L}_M$ .

The first rule (*R1*) below states that mental state formulas that ‘have the form of a classical tautology’ (like  $(B\phi \vee \neg B\phi)$  and  $G\phi_1 \rightarrow (B\phi_2 \rightarrow G\phi_1)$ ), are also derivable in  $\vdash_M$ . By the necessitation rule (*R2*), an agent believes all classical tautologies. Then, (*A1*) expresses that the belief modality distributes over implication. This implies that the beliefs of an agent are closed under logical consequence. Finally, *A2* states that the beliefs of an agent are consistent. In essence, the belief operator thus satisfies the properties of the system **KD** (see [9, 21]). Although in its current presentation, our language does not allow for nested (belief-) operators, from [21, Section 1.7] we conclude that we may assume *as if* our agent has positive  $(B\phi \rightarrow BB\phi)$  and negative  $(\neg B\phi \rightarrow B\neg B\phi)$  introspective properties: every formula in the system **KD45** (which is **KD** together with the two mentioned properties) is equivalent to a formula without nestings of operators.

Axiom *A4* below, is a consequence of the constraint on mental states and expresses that if an agent believes  $\phi$  it does not have a goal to achieve  $\phi$ . As a consequence, an agent cannot have a goal to achieve a tautology:  $\neg G\text{true}$ . An agent also does not have inconsistent goals (*A3*), that is,  $\neg G\text{false}$  is an axiom (see Table 2). Finally, the conditions that allow to conclude that the agent has a (sub)goal  $\psi$  are that the agent has a goal  $\phi$  that logically entails  $\psi$  and that

$R1$	if $\varphi$ is an instantiation of a classical tautology, then $\vdash_M \varphi$
$R2$	$\vdash_C \phi \Rightarrow \vdash_M B\phi$ , for $\phi \in \mathcal{L}$
$A1$	$\vdash_M B(\phi \rightarrow \psi) \rightarrow (B\phi \rightarrow B\psi)$
$A2$	$\vdash_M \neg B\text{false}$

Table 1: Properties of Beliefs

the agent does not believe that  $\psi$  is the case. Axiom  $A5$  below then allows to conclude that  $G\psi$  holds. From now on, for any mental state formula  $\varphi$ ,  $\vdash_M \varphi$  means that there is a derivation of  $\varphi$  using the proof rules  $R1$  and  $R2$  and the axioms  $A1 - A6$ . If  $\Delta$  is a set of mental state formulas from  $\mathcal{L}_M$ , then  $\Delta \vdash_M \varphi$  means that there is a derivation of  $\varphi$  using the rules and axioms mentioned, and the formulas of  $\Delta$  as premises.

$A3$	$\vdash_M \neg G\text{false}$
$A4$	$\vdash_M B\phi \rightarrow \neg G\phi$
$A5$	$\vdash_C \phi \rightarrow \psi \Rightarrow \vdash_M \neg B\psi \rightarrow (G\phi \rightarrow G\psi)$

Table 2: Properties of Goals

The goal modality is a weak logical operator. For example, the goal modality does not distribute over implication. A counter example is provided by the goal base that is generated from  $\{p, p \rightarrow q\}$ . The consequences of goals are only computed *locally*, from individual goals. But even from the goal base  $\{p \wedge (p \rightarrow q)\}$  one cannot conclude that  $q$  is a goal, since this conclusion is blocked in a mental state in which  $q$  is already believed. Deriving only consequences of goals locally ensures that from the fact that  $G\phi$  and  $G\psi$  hold, it is not possible to conclude that  $G(\phi \wedge \psi)$ . This reflects the fact that individual goals cannot be added to a single bigger goal; recall that two individual goals may be inconsistent ( $G\phi \wedge G\neg\phi$  is satisfiable) in which case taking the conjunction would lead to an inconsistent goal. In sum, most of the usual problems that many logical operators for motivational attitudes suffer from do not apply to our  $G$  operator (cf. also [22]). On the other hand, the last property of Lemma 2.4 justifies to call  $G$  a logical, and not just a syntactical operator:

**Lemma 2.4**

- $\not\models_M G(\phi \rightarrow \psi) \rightarrow (G\phi \rightarrow G\psi)$ ,
- $\not\models_M G(\phi \wedge (\phi \rightarrow \psi)) \rightarrow G\psi$ ,
- $\not\models_M (G\phi \wedge G\psi) \rightarrow G(\phi \wedge \psi)$
- $\models_C (\varphi \leftrightarrow \psi) \Rightarrow \models_M (G\varphi \leftrightarrow G\psi)$

One finds a similar quest for such weak operators in *awareness logics* for doxastic and epistemic modalities, see e.g. [8, 31]. As agents do not want all the side-effects of their goals, being limited reasoners they also do not always adopt all the logical consequences of their belief or knowledge. However, the question remains whether modal logic is *the* formal tool to reason with and about goals. Allowing explicitly for mutually inconsistent goals, our treatment of goals resides in the landscape of *paraconsistent logic* (cf. [23]). One might even go a step further and explore to use *linear logic* ([11] to reason about goals, enabling to have the same goal more than once, and to model process and resource use in a fine-tuned way. We will not pursue the different options for logics of goals in this paper.

**Theorem 2.5** (*Soundness and Completeness of  $\vdash_M$* )

For any  $\varphi \in \mathcal{L}_M$ , we have

$$\vdash_M \varphi \Leftrightarrow \models_M \varphi$$

**Proof.** We leave it for the reader to check soundness (i.e., the ‘ $\Rightarrow$ ’-direction). Here, A1 and A2 are immediate consequences of the definition of a belief as a consequence from a given consistent set, A3 follows from condition (ii) of Definition 2.1, A4 from property (i) and A5 from (iii) of that same definition. For completeness, assume that  $\not\vdash_M \varphi$ . Then  $\neg\varphi$  is consistent, and we will construct a mental state  $\langle \Sigma, \Gamma \rangle$  that verifies  $\varphi$ . First, we build a maximal  $\vdash_M$ -consistent set  $\Delta$  with  $\neg\varphi \in \Delta$ . This  $\Delta$  can be split in a set  $\Sigma$  and a set  $\Gamma$  as follows:  $\Sigma = \{\phi \mid B\phi \in \Delta\}$  and  $\Gamma = \{\phi \mid G\phi \in \Delta\}$ . We now prove two properties of  $\langle \Sigma, \Gamma \rangle$ :

1.  $\langle \Sigma, \Gamma \rangle$  is a mental state
2.  $\langle \Sigma, \Gamma \rangle$  satisfies the following coincidence property:

$$\text{for all } \chi \in \mathcal{L}_m : \langle \Sigma, \Gamma \rangle \models_M \chi \Leftrightarrow \chi \in \Delta$$

The proofs for these claims are as follows:

1. We must show that  $\langle \Sigma, \Gamma \rangle$  satisfies the properties of Definition 2.1. Obviously,  $\Sigma$  is classically consistent, since otherwise we would have  $B\perp$  in the  $\vdash_M$ -consistent set  $\Delta$ , which is prohibited by axiom A2. Also, by axiom A3, no  $\gamma \in \Gamma$  is equivalent to  $\perp$ . We now show that no  $\gamma \in \Gamma$  is classically entailed by  $\Sigma$ . Suppose that we would have that  $\sigma_1, \dots, \sigma_n \vdash_C \gamma$ , for certain  $\sigma_1, \dots, \sigma_n \in \Sigma$  and  $\gamma \in \Gamma$ . Then, by construction of  $\Sigma$  and  $\Gamma$ , the formulas  $B\sigma_1, \dots, B\sigma_n, G\gamma$  all are members of the maximal  $\vdash_M$ -consistent set  $\Delta$ . Since  $\sigma_1, \dots, \sigma_n \vdash_C \gamma$ , by the deduction theorem for  $\vdash_C$ , R2 and A2 we conclude  $\vdash_M (B\sigma_1, \dots, B\sigma_n) \rightarrow B\gamma$ . But this means that both  $B\gamma$  and  $G\gamma$  are members of  $\Delta$ , which is prohibited by axiom A4. Finally, we show (iii) of Definition 2.1. Suppose  $\gamma \in \Gamma$ ,  $\models_C \gamma \rightarrow \gamma'$  and that  $\gamma'$  is consistent, and not classically entailed by  $\Sigma$ . We have to  $\gamma' \in \Gamma$ , and this is immediately guaranteed by axiom A5.

2. The base case for the second claim is about  $B\phi$  and  $G\phi$ , with  $\phi \in \mathcal{L}$ . We have  $\langle \Sigma, \Gamma \rangle \models_M B\phi$  iff  $\Sigma \models_C \phi$  iff, by definition of  $\Sigma$ ,  $\{\sigma \mid B\sigma \in \Delta\} \models_C \phi$ . Using compactness and the deduction theorem for classical logic, we find  $\vdash_C \sigma_1 \wedge \dots \wedge \sigma_n \rightarrow \phi$ , for some propositional formulas  $\sigma_1, \dots, \sigma_n$ . By the rule *R2* we conclude  $\vdash_M B(\sigma_1 \wedge \dots \wedge \sigma_n) \rightarrow \phi$ . By *A1*, this is equivalent to  $\vdash_M (B\sigma_1 \wedge \dots \wedge B\sigma_n) \rightarrow B\phi$  and, since all the  $B\sigma_i (i \leq n)$  are members of  $\Delta$ , we have  $B\phi \in \Delta$ . For the other base case, consider  $\langle \Sigma, \Gamma \rangle \models_M G\phi$ , which, using the truth-definition for  $G$ , holds iff  $\gamma \in \Gamma$ . By definition of  $\Gamma$ , this means that  $G\gamma \in \Delta$ , which was to be proven. The cases for negation and conjunction follow immediately from this. Hence, in particular, we have  $\langle \Sigma, \Gamma \rangle \models_M \neg\varphi$ , and thus  $\not\models_M \varphi$ .

## 2.2 GOAL Agents

A third basic concept in GOAL is that of an agent *capability*. The capabilities of an agent consist of a set of so called *basic actions*. The effects of executing such a basic action are reflected in the beliefs of the agent and therefore a basic action is taken to be a belief update on the agent's beliefs. A basic action thus is a *mental state transformer*. Two examples of agent capabilities are the actions  $\text{ins}(\phi)$  for inserting  $\phi$  in the belief base and  $\text{del}(\phi)$  for removing  $\phi$  from the belief base. Agent capabilities directly affect the belief base of the agent and not its goals, but because of the constraints on mental states they may as a side effect modify the current goals. For the purpose of modifying the goals of the agent, two special actions  $\text{adopt}(\phi)$  and  $\text{drop}(\phi)$  are introduced to respectively adopt a new goal or drop some old goals. We write  $Bcap$  and use it to denote the set of all belief update capabilities of an agent.  $Bcap$  thus does not include the two special actions for goal updating  $\text{adopt}(\phi)$  and  $\text{drop}(\phi)$ . The set of all capabilities is then defined as  $Cap = Bcap \cup \{\text{adopt}(\phi), \text{drop}(\phi) \mid \phi \in \mathcal{L}\}$ . Individual capabilities are denoted by  $\mathbf{a}$ .

The set of basic actions or capabilities associated with an agent determines what an agent *is able to do*. It does not specify *when* such a capability should be exercised and when performing a basic action is to the agent's advantage. To specify such conditions, the notion of a *conditional action* is introduced. A conditional action consists of a mental state condition expressed by a mental state formula and a basic action. The mental state condition of a conditional action states the conditions that must hold for the action to be selected. Conditional actions are denoted by the symbol  $b$  throughout this paper.

### Definition 2.6 (*conditional action*)

A *conditional action* is a pair  $\varphi \rightarrow do(\mathbf{a})$  such that  $\varphi \in \mathcal{L}_M$  and  $\mathbf{a} \in Cap$ .

Informally, a conditional action  $\varphi \rightarrow do(\mathbf{a})$  means that if the mental condition  $\varphi$  holds, then the agent may consider doing basic action  $\mathbf{a}$ . Of course, if the mental state condition holds in the current state, the action  $\mathbf{a}$  can only be successfully executed if the action is *enabled*, that is, only if its precondition holds.



A GOAL agent consists of a specification of an *initial mental state* and a set of conditional actions.

**Definition 2.7** (*GOAL agent*)

A *GOAL agent* is a triple  $\langle \Pi, \Sigma_0, \Gamma_0 \rangle$  where  $\Pi$  is a non-empty set of conditional actions, and  $\langle \Sigma_0, \Gamma_0 \rangle$  is the initial mental state.

### 2.3 The Operational Semantics of GOAL

One of the key ideas in the semantics of GOAL is to incorporate into the semantics a particular *commitment strategy* (cf. [26, 5]). The semantics is based on a particularly simple and transparent commitment strategy, called *blind commitment*. An agent that acts according to a blind commitment strategy drops a goal if and only if it believes that that goal has been achieved. By incorporating this commitment strategy into the semantics of GOAL, a *default* commitment strategy is built into agents. It is, however, only a default strategy and a programmer can overwrite this default strategy by means of the **drop** action. It is not possible, however, to adopt a goal  $\phi$  in case the agent believes that  $\phi$  is already achieved.

The semantics of action execution should now be defined in conformance with this basic commitment principle. Recall that the basic capabilities of an agent were interpreted as belief updates. Because of the default commitment strategy, there is a relation between beliefs and goals, however, and we should extend the belief update associated with a capability to a mental state transformer that updates beliefs as well as goals according to the blind commitment strategy. To get started, we thus assume that some specification of the belief update semantics of all capabilities - except for the two special actions **adopt** and **drop** which only update goals - is given. Our task is, then, to construct a mental state transformer semantics from this specification for each action. That is, we must specify how a basic action updates the complete current mental state of an agent starting with a specification of the belief update associated with the capability only.

From the default blind commitment strategy, we conclude that if a basic action **a** - different from an **adopt** or **drop** action - is executed, then a goal is dropped *only if* the agent *believes* that the goal has been accomplished after doing **a**. The revision of goals thus is based on the beliefs of the agent. The beliefs of an agent represent all the information that is available to an agent to decide whether or not to drop or adopt a goal. So, in case the agent believes that a goal has been achieved by performing some action, then this goal must be removed from the current goals of the agent. Besides the default commitment strategy, only the two special actions **adopt** and **drop** can result in a change to the goal base.

The initial specification of the belief updates associated with the capabilities *Bcap* is formally represented by a partial function  $\mathcal{T}$  of type  $Bcap \times \wp(\mathcal{L}) \rightarrow \wp(\mathcal{L})$ .  $\mathcal{T}(\mathbf{a}, \Sigma)$  returns the result of updating belief base  $\Sigma$  by performing action **a**. The fact that  $\mathcal{T}$  is a *partial* function represents the fact that an action may

not be *enabled* or executable in some belief states. The mental state transformer function  $\mathcal{M}$  is derived from the semantic function  $\mathcal{T}$  and also is a partial function. As explained,  $\mathcal{M}(\mathbf{a}, \langle \Sigma, \Gamma \rangle)$  removes any goals from the goal base  $\Gamma$  that have been achieved by doing  $\mathbf{a}$ . The function  $\mathcal{M}$  also defines the semantics of the two special actions **adopt** and **drop**. An **adopt**( $\phi$ ) action adds  $\phi$  to the goal base if  $\phi$  is consistent and  $\phi$  is not believed to be the case. A **drop**( $\phi$ ) action removes every goal that entails  $\phi$  from the goal base. As an example, consider the two extreme cases: **drop**(false) removes no goals, whereas **drop**(true) removes all current goals.

**Definition 2.8** (*mental state transformer  $\mathcal{M}$* )

Let  $\langle \Sigma, \Gamma \rangle$  be a mental state, and  $\mathcal{T}$  be a partial function that associates belief updates with agent capabilities. Then the partial function  $\mathcal{M}$  is defined by:

$$\begin{aligned} \mathcal{M}(\mathbf{a}, \langle \Sigma, \Gamma \rangle) &= \langle \mathcal{T}(\mathbf{a}, \Sigma), \Gamma \setminus \{\psi \in \Gamma \mid \mathcal{T}(\mathbf{a}, \Sigma) \models_C \psi\} \rangle \\ &\quad \text{for } \mathbf{a} \in Bcap, \text{ if } \mathcal{T}(\mathbf{a}, \Sigma) \text{ is defined} \\ \mathcal{M}(\mathbf{a}, \langle \Sigma, \Gamma \rangle) &\text{ is undefined for } \mathbf{a} \in Bcap \text{ if } \mathcal{T}(\mathbf{a}, \Sigma) \text{ is undefined} \\ \mathcal{M}(\mathbf{drop}(\phi), \langle \Sigma, \Gamma \rangle) &= \langle \Sigma, \Gamma \setminus \{\psi \in \Gamma \mid \psi \models_C \phi\} \rangle \\ \mathcal{M}(\mathbf{adopt}(\phi), \langle \Sigma, \Gamma \rangle) &= \langle \Sigma, \Gamma \cup \{\phi' \mid \Sigma \not\models_M \phi' \ \& \ \models_C \phi \rightarrow \phi'\} \rangle \\ &\quad \text{if } \not\models_C \neg\phi \text{ and } \Sigma \not\models_C \phi \\ \mathcal{M}(\mathbf{adopt}(\phi), \langle \Sigma, \Gamma \rangle) &\text{ is undefined if } \Sigma \models_C \phi \text{ or } \models_C \neg\phi \end{aligned}$$

The semantic function  $\mathcal{M}$  maps an agent capability and a mental state to a new mental state. The capabilities of an agent are thus interpreted as *mental state transformers* by  $\mathcal{M}$ . Although it is not allowed to adopt a goal  $\phi$  that is inconsistent - an **adopt**(false) is not enabled - there is no check on the global consistency of the goal base of an agent built into the semantics. This means that it is allowed to adopt a new goal which is inconsistent with another goal present in the goal base. For example, if the current goal base  $\Gamma$  contains  $p$ , it is legal to execute the action **adopt**( $\neg p$ ) resulting in a new goal base containing  $p, \neg p$ , (if  $\neg p$  was not already believed). Although inconsistent goals cannot be achieved at the same time, they may be achieved in some temporal order. Individual goals in the goal base, however, are required to be consistent. Thus, whereas local consistency is required (i.e. individual goals must be consistent), global consistency of the goal base is not required.

The second idea incorporated into the semantics concerns the *selection of conditional actions*. A conditional action  $\varphi \rightarrow do(\mathbf{a})$  may specify conditions on the beliefs as well as conditions on the goals of an agent. As is usual, conditions on the beliefs are taken as a precondition for action execution: only if the agent's current beliefs entail the belief conditions associated with  $\varphi$  the agent will select  $\mathbf{a}$  for execution. The goal condition, however, is used in a different way. It is used as a means for the agent to determine whether or not the action will help bring about a particular goal of the agent. In short, the goal condition specifies where the action is good for. This does not mean that the action necessarily establishes the goal immediately, but rather may be taken as an indication that the action is helpful in bringing about a particular state of affairs.

In the definition below, we assume that the action component  $\Pi$  of an agent  $\langle \Pi, \Sigma_0, \Gamma_0 \rangle$  is fixed. The execution of an action gives rise to a *computation*

*step* formally denoted by the transition relation  $\xrightarrow{b}$  where  $b$  is the conditional action executed in the computation step. More than one computation step may be possible in a current state and the step relation  $\longrightarrow$  thus denotes a *possible* computation step in a state. A computation step updates the current state and yields the next state of the computation. Note that because  $\mathcal{M}$  is a partial function, a conditional action can only be successfully executed if both the condition is satisfied and the basic action is enabled.

**Definition 2.9** (*action selection*)

Let  $\langle \Sigma, \Gamma \rangle$  be a mental state and  $b = \varphi \rightarrow do(a) \in \Pi$ . Then, as a rule, we have: If

- the mental condition  $\varphi$  holds in  $\langle \Sigma, \Gamma \rangle$ , i.e.  $\langle \Sigma, \Gamma \rangle \models \varphi$ , and
- $a$  is enabled in  $\langle \Sigma, \Gamma \rangle$ , i.e.  $\mathcal{M}(a, \langle \Sigma, \Gamma \rangle)$  is defined,

then  $\langle \Sigma, \Gamma \rangle \xrightarrow{b} \mathcal{M}(a, \langle \Sigma, \Gamma \rangle)$  is a possible computation step. The relation  $\longrightarrow$  is the smallest relation closed under this rule.

Now, the semantics of GOAL agents is derived directly from the operational semantics and the computation step relation  $\longrightarrow$ . The meaning of a GOAL agent consists of a set of so called *traces*. A trace is an infinite computation sequence of consecutive mental states interleaved with the actions that are scheduled for execution in each of those mental states. The fact that a conditional action is scheduled for execution in a trace does not mean that it is also enabled in the particular state for which it has been scheduled. In case an action is scheduled but not enabled, the action is simply skipped and the resulting state is the same as the state before. In other words, enabledness is not a criterion for selection, but rather it decides whether something is happening in a state, once selected.

**Definition 2.10** (*trace*)

A trace  $s$  is an infinite sequence  $s_0, b_0, s_1, b_1, s_2, \dots$  such that  $s_i$  is a mental state,  $b_i$  is a conditional action, and for every  $i$  we have:  $s_i \xrightarrow{b_i} s_{i+1}$ , or  $b_i$  is not enabled in  $s_i$  and  $s_i = s_{i+1}$ .

An important assumption in the semantics for GOAL is a *fairness* assumption. Fairness assumptions concern the fair selection of actions during the execution of a program. In our case, we make a *weak fairness* assumption [20]. A trace is weakly fair if it is not the case that an action is always enabled from some point in time on but is never selected for execution. This weak fairness assumption is built into the semantics by imposing a constraint on traces. By definition, a *fair trace* is a trace in which each of the actions is scheduled infinitely often. In a fair trace, there always will be a future time point at which an action is scheduled (considered for execution) and by this scheduling policy a fair trace implements the weak fairness assumption. However, note that the fact that an action is scheduled does not mean that the action also is enabled

(and therefore, the selection of the action may result in an idle step which does not change the state).

The meaning of a GOAL agent now is defined as the set of fair traces in which the initial state is the initial mental state of the agent and each of the steps in the trace corresponds to the execution of a conditional action or an idle transition.

**Definition 2.11** (*meaning of a GOAL agent*)

The meaning of a GOAL agent  $\langle \Pi, \Sigma_0, \Gamma_0 \rangle$  is the set of *fair* traces  $S$  such that for  $s \in S$  we have  $s_0 = \langle \Sigma_0, \Gamma_0 \rangle$ .

## 2.4 Mental States and Enabledness

We formally said that a capability  $\mathbf{a} \in \text{Cap}$  is *enabled* in a mental state  $\langle \Sigma, \Gamma \rangle$  in case  $\mathcal{M}(\mathbf{a}, \langle \Sigma, \Gamma \rangle)$  is defined. This definition implies that a belief update capability  $\mathbf{a} \in \text{Bcap}$  is enabled if  $\mathcal{T}(\mathbf{a}, \Sigma)$  is defined. Let us assume that this only depends on the action  $\mathbf{a}$ —this seems reasonable, since a paradigm like AGM ([1]) only requires that a revision with  $\varphi$  fails iff  $\varphi$  is classically inconsistent, whereas expansions and contractions succeed for all  $\varphi$ , hence the question whether such an operation is enabled does not depend on the current beliefs. A conditional action  $b$  is *enabled* in a mental state  $\langle \Sigma, \Gamma \rangle$  if there are  $\Sigma', \Gamma'$  such that  $\langle \Sigma, \Gamma \rangle \xrightarrow{b} \langle \Sigma', \Gamma' \rangle$ . Note that if a capability  $\mathbf{a}$  is not enabled, a conditional action  $\varphi \rightarrow \text{do}(\mathbf{a})$  is also not enabled. The special predicate *enabled* is introduced to denote that a capability  $\mathbf{a}$  or conditional action  $b$  is enabled (denoted by *enabled*( $\mathbf{a}$ ) respectively *enabled*( $b$ )).

The relation between the enabledness of capabilities and conditional actions is stated in the next table together with the fact that  $\text{drop}(\phi)$  is always enabled and a proof rule for deriving *enabled*( $\text{adopt}(\phi)$ ). Let  $\mathcal{L}_{ME}$  be the language obtained by Boolean combinations of mental state formulas and enabledness formulas. We denote derivability in the system for this language by  $\vdash_{ME}$ . Then,  $\vdash_{ME}$  consists of the axioms and rules for  $\vdash_M$ , plus

<i>E1</i>	$\text{enabled}(\varphi \rightarrow \text{do}(\mathbf{a})) \leftrightarrow (\varphi \wedge \text{enabled}(\mathbf{a})),$
<i>E2</i>	$\text{enabled}(\text{drop}(\phi)),$
<i>R3</i>	$\not\models_C \neg\phi \Rightarrow \vdash_{ME} \neg \text{B}\phi \leftrightarrow \text{enabled}(\text{adopt}(\phi))$
<i>R4</i>	$\models_C \neg\phi \Rightarrow \vdash_{ME} \neg \text{enabled}(\text{adopt}(\phi))$
<i>R5</i>	$\vdash_{ME} \text{enabled}(\mathbf{a})$ if $\mathcal{T}(\mathbf{a}, \cdot)$ is defined ( $\mathbf{a} \in \text{Bcap}$ )

Table 3: Enabledness

Rule *R5* enforces that we better write  $\vdash_{ME}^{\mathcal{T}}$  given a belief revision function  $\mathcal{T}$ , but in the sequel we will suppress this  $\mathcal{T}$ . The semantics  $\models_{ME}$  for  $\mathcal{L}_{ME}$  is based on truth in pairs  $\langle \Sigma, \Gamma \rangle, \mathcal{T}$ , where  $\langle \Sigma, \Gamma \rangle$  is a mental state and  $\mathcal{T}$  a partial function for belief updates. For formulas of the format  $\text{B}\varphi$  and  $\text{G}\varphi$ , we just use the mental state and Definition 2.3 to determine their truth. For enabledness formulas, we have the following:

**Definition 2.12** (Truth of enabledness)

- $\langle \Sigma, \Gamma \rangle, \mathcal{T} \models_{ME} \text{enabled}(\mathbf{a})$  iff  $\mathcal{T}(\mathbf{a}, \Sigma)$  is defined
- $\langle \Sigma, \Gamma \rangle, \mathcal{T} \models_{ME} \text{enabled}(\text{drop}(\phi))$  iff true
- $\langle \Sigma, \Gamma \rangle, \mathcal{T} \models_{ME} \text{enabled}(\text{adopt}(\phi))$  iff  $\not\models_C \neg\phi$  and  $\langle \Sigma, \Gamma \rangle, \mathcal{T} \models_{ME} \neg \mathbf{B}\phi$
- $\langle \Sigma, \Gamma \rangle, \mathcal{T} \models_{ME} \text{enabled}(\varphi \rightarrow \text{do}(\mathbf{a}))$  iff  $\langle \Sigma, \Gamma \rangle, \mathcal{T} \models_{ME} \phi$  and at the same time  $\langle \Sigma, \Gamma \rangle, \mathcal{T} \models_{ME} \text{enabled}(\mathbf{a})$

Note that we can summarize this definition to:

- $\langle \Sigma, \Gamma \rangle, \mathcal{T} \models_{ME} \text{enabled}(\mathbf{a})$  iff  $\mathcal{M}(\mathbf{a}, \langle \Sigma, \Gamma \rangle)$  is defined for  $\mathbf{a} \in \text{Cap}$ ,
- $\langle \Sigma, \Gamma \rangle, \mathcal{T} \models_{ME} \text{enabled}(b)$  iff  $\models_{ME} \varphi$  and there are  $\Sigma', \Gamma'$  such that  $\langle \Sigma, \Gamma \rangle \xrightarrow{b} \langle \Sigma', \Gamma' \rangle$  for conditional actions where  $b = \varphi \rightarrow \text{do}(\mathbf{a})$ .

**Theorem 2.13** (Soundness and Completeness of  $\vdash_{ME}$ )

We have, for all formulas  $\varphi$  in  $\mathcal{L}_{ME}$ ,

$$\vdash_{ME} \varphi \text{ iff } \models_{ME} \varphi$$

**Proof.** Again, checking soundness is straightforward and left to the reader. For the converse, we have to make a complexity measure explicit for  $\mathcal{L}_{ME}$ -formulas, along which the induction can proceed. It suffices to stipulate that the complexity of  $\text{enabled}(\psi \rightarrow \text{do}(\mathbf{a}))$  is greater than that of  $\mathbf{B}\psi$  and  $\text{enabled}(\mathbf{a})$ . Furthermore, the complexity of  $\text{enabled}(\text{adopt}(\phi))$  is greater than that of  $(\neg)\mathbf{B}\phi$ . Now, suppose that  $\not\models_{ME} \varphi$ , i.e.,  $\neg\varphi$  is consistent. Note that the language  $\mathcal{L}_{ME}$  is countable, so that we can by enumeration, extend  $\{\neg\varphi\}$  to a maximal  $\vdash_{ME}$ -consistent set  $\Delta$ . From this  $\Delta$ , we distill a pair  $\langle \Sigma, \Gamma \rangle, \mathcal{T}$  as follows:  $\Sigma = \{\varphi \mid \mathbf{B}\varphi \in \Delta\}$ ,  $\Gamma = \{\varphi \mid \mathbf{G}\varphi \in \Delta\}$ , and  $\mathcal{T}(\mathbf{a}, \Sigma)$  is defined iff  $\text{enabled}(\mathbf{a}) \in \Delta$ , for any belief capability  $\mathbf{a}$ . We claim, for all  $\chi \in \mathcal{L}_{ME}$ :

$$\langle \Sigma, \Gamma \rangle, \mathcal{T} \models \chi \text{ iff } \chi \in \Delta$$

For formulas of type  $\mathbf{B}\psi$  and  $\mathbf{G}\psi$  this is easily seen. Let us check it for enabledness formulas.

- $\chi = \text{enabled}(\mathbf{a})$ , with  $\mathbf{a}$  a belief capability. By construction of  $\mathcal{T}$ , the result immediately holds
- $\chi = \text{enabled}(\text{drop}(\phi))$ . By construction of  $\Delta$ , every  $\text{enabled}(\text{drop}(\phi))$  is an element of  $\Delta$  (because of axiom E2), and also, every such formula is true in  $\langle \Sigma, \Gamma \rangle, \mathcal{T}$ .
- $\chi = \text{enabled}(\text{adopt}(\phi))$ . Suppose  $\langle \Sigma, \Gamma \rangle, \mathcal{T} \models_{ME} \chi$ . Then,  $\not\models \neg\phi$  and  $\langle \Sigma, \Gamma \rangle, \mathcal{T} \not\models_{ME} \mathbf{B}\phi$ . By the induction hypothesis, we have that  $\mathbf{B}\phi \notin \Delta$ , hence  $\neg\mathbf{B}\phi \in \Delta$ , and, by R3,  $\text{enabled}(\text{adopt}(\phi)) \in \Delta$ . For the converse,

suppose  $enabled(adopt(\phi)) \in \Delta$ . Then (by  $R4$ ), we cannot have that  $\models_C \neg\phi$ . Hence,  $\not\models_C \neg\phi$ , and by  $R3$ , we also have  $\neg B\phi \in \Delta$  and hence, by applying the induction hypothesis,  $\langle \Sigma, \Gamma \rangle, \mathcal{T} \models_C \neg B\phi$ . Since  $R3$  is a sound rule, we finally conclude that  $\langle \Sigma, \Gamma \rangle, \mathcal{T} \models_{ME} enabled(adopt(\phi))$ .

- $\chi = enabled(\psi \rightarrow a)$ . We can write this as  $\psi \wedge enabled(a)$  and then use the induction hypothesis.

### 3 A Personal Assistant Example

In this section, we give an example to show how the programming language GOAL can be used to program agents. The example concerns a shopping agent that is able to buy books on the Internet on behalf of the user. The example provides for a simple illustration of how the programming language works. The agent in our example uses a standard procedure for buying a book. It first goes to a bookstore, in our case Am.com. At the web site of Am.com it searches for a particular book, and if the relevant page with the book details shows up, the agent puts the book in its shopping cart. In case the shopping cart of the agent contains some items, it is allowed to buy the items on behalf of the user. The idea is that the agent adopts a goal to buy a book if the user instructs it to do so.

The set of capabilities  $Bcap$  of the agent is defined by

$$\{goto\_website(site), search(book), put\_in\_shopping\_cart(book), pay\_cart\}$$

The capability  $goto\_website(site)$  goes to the selected web page  $site$ . In our example, relevant web pages are the home page of the user, the main page of Am.com, web pages with information about books to buy, and a web page that shows the current items in the shopping cart of the agent. The capability  $search(book)$  is an action that can be selected at the main page of Am.com and selects the web page with information about  $book$ . The action  $put\_in\_shopping\_cart(book)$  can be selected on the page concerning  $book$  and puts  $book$  in the cart; a new web page called *ContentCart* shows up showing the content of the cart. Finally, in case the cart is not empty the action  $pay\_cart$  can be selected to pay for the books in the cart.

In the program text below, we assume that  $book$  is a variable referring to the specifics of the book the user wants to buy (in the example, we use variables as a means for abbreviation; variables should be thought of as being instantiated with the relevant arguments in such a way that predicates with variables reduce to propositions). The initial beliefs of the agent are that the current web page is the home page of the user, and that it is not possible to be on two different web pages at the same time. We also assume that the user has provided the agent with the goals to buy *The Intentional Stance* by Daniel Dennett and *Intentions, Plans, and Practical Reason* by Michael Bratman.

$$\begin{aligned}
\Pi = \{ & \text{B}(\text{current\_website}(\text{hpage}(\text{user})) \vee \text{current\_website}(\text{ContentCart})) \\
& \wedge \text{G}(\text{bought}(\text{book})) \rightarrow \text{do}(\text{goto\_website}(\text{Am.com})), \\
& \text{B}(\text{current\_website}(\text{Am.com})) \wedge \neg \text{B}(\text{in\_cart}(\text{book})) \wedge \\
& \text{G}(\text{bought}(\text{book})) \rightarrow \text{do}(\text{search}(\text{book})), \\
& \text{B}(\text{current\_website}(\text{book})) \wedge \text{G}(\text{bought}(\text{book})) \\
& \rightarrow \text{do}(\text{put\_in\_shopping\_cart}(\text{book})), \\
& \text{B}(\text{in\_cart}(\text{book})) \wedge \text{G}(\text{bought}(\text{book})) \rightarrow \text{do}(\text{pay\_cart}) \}, \\
\Sigma_0 = \{ & \text{current\_webpage}(\text{hpage}(\text{user})), \\
& \forall s, s' ((s \neq s' \wedge \text{current\_webpage}(s)) \rightarrow \neg \text{current\_webpage}(s')) \}, \\
\Gamma_0 = \{ & \text{bought}(\text{The Intentional Stance}) \\
& \wedge \text{bought}(\text{Intentions, Plans and Practical Reason}) \} \\
& \text{GOAL Shopping Agent}
\end{aligned}$$

Some of the details of this program will be discussed in the sequel, when we prove some properties of the program. The agent basically follows the recipe for buying a book outlined above. For now, however, just note that the program is quite flexible, even though the agent more or less executes a fixed recipe for buying a book. The flexibility results from the agent's knowledge state and the non-determinism of the program. In particular, the ordering in which the actions are performed by the agent - which book to find first, buy a book one at a time or both in the same shopping cart, etc. is not determined by the program. The scheduling of these actions thus is not fixed by the program, and might be fixed arbitrarily on a particular agent architecture used to run the program.

## 4 Logic for GOAL

On top of the language GOAL and its semantics, we now construct a temporal logic to prove properties of GOAL agents. The logic is similar to other temporal logics but its semantics is derived from the operational semantics for GOAL. Moreover, the logic incorporates the belief and goal modalities used in GOAL agents. We first informally discuss the use of Hoare triples for the specification of actions. In Section 4.3 we give a sound and complete system for such triples. These Hoare triples play an important role in the programming logic since it can be shown that temporal properties of agents can be proven by means of proving Hoare triples for actions only. Finally, in 4.4 the language for expressing temporal properties and its semantics is defined and the fact that certain classes of interesting temporal properties can be reduced to properties of actions, expressed by Hoare triples, is proven.

### 4.1 Hoare Triples

The specification of basic actions provides the basis for the programming logic, and, as we will show below, is all we need to prove properties of agents. Because

they play such an important role in the proof theory of GOAL, the specification of the basic agent capabilities requires special care. In the proof theory of GOAL, Hoare triples of the form  $\{\varphi\} b \{\psi\}$ , where  $\varphi$  and  $\psi$  are *mental state formulas*, are used to specify actions. The use of Hoare triples in a formal treatment of traditional assignments is well-understood [2]. Because the agent capabilities of GOAL agents are quite different from assignment actions, however, the traditional predicate transformer semantics is not applicable. GOAL agent capabilities are mental state transformers and, therefore, we require more extensive basic action theories to formally capture the effects of such actions. Hoare triples are used to specify the *postconditions* and the *frame conditions* of actions. The postconditions of an action specify the effects of an action whereas the frame conditions specify what is not changed by the action. Axioms for the predicate *enabled* specify the preconditions of actions.

The formal semantics of a Hoare triple for conditional actions is derived from the semantics of a GOAL agent and is defined relative to the set of traces  $S_A$  associated with the GOAL agent  $A$ . A Hoare triple for conditional actions thus expresses a property of an agent and not just a property of an action. The semantics of the basic capabilities are assumed to be fixed, however, and are not defined relative to an agent.

**Definition 4.1** (*semantics of Hoare triples for basic actions*)

A Hoare triple for basic capabilities  $\{\varphi\} a \{\psi\}$  means that for all  $\Sigma, \Gamma$

- $\langle \Sigma, \Gamma \rangle \models \varphi \wedge \text{enabled}(a) \Rightarrow \mathcal{M}(a, \langle \Sigma, \Gamma \rangle) \models \psi$ , and
- $\langle \Sigma, \Gamma \rangle \models \varphi \wedge \neg \text{enabled}(a) \Rightarrow \langle \Sigma, \Gamma \rangle \models \psi$ .

To explain this definition, note that we made a case distinction between states in which the basic action is enabled and in which it is not enabled. In case the action is enabled, the postcondition  $\psi$  of the Hoare triple  $\{\varphi\} a \{\psi\}$  should be evaluated in the next state resulting from executing action  $a$ . In case the action is not enabled, however, the postcondition should be evaluated in the same state because a failed attempt to execute action  $a$  is interpreted as an idle step in which nothing changes.

Hoare triples for conditional actions are interpreted *relative to the set of traces* associated with the GOAL agent of which the action is a part. Below, we write  $\varphi[s_i]$  to denote that a mental state formula  $\varphi$  holds in state  $s_i$ .

**Definition 4.2** (*semantics of Hoare triples for conditional actions*)

Given an agent  $A$ , a Hoare triple for conditional actions  $\{\varphi\} b \{\psi\}$  (for  $A$ ) means that for all traces  $s \in S_A$  and  $i$ , we have that

$$(\varphi[s_i] \wedge b = b_i \in s) \Rightarrow \psi[s_{i+1}]$$

where  $b_i \in s$  means that action  $b_i$  is taken in state  $i$  of trace  $s$ .

Of course, there is a relation between the execution of basic actions and that of conditional actions, and therefore there also is a relation between the two types of Hoare triples. The following lemma makes this relation precise.



**Lemma 4.3** Let  $A$  be a GOAL agent and  $S_A$  be the meaning of  $A$ . Suppose that we have  $\{\varphi \wedge \psi\} \mathbf{a} \{\varphi'\}$  and  $S_A \models (\varphi \wedge \neg\psi) \rightarrow \varphi'$ . Then we also have  $\{\varphi\} \psi \rightarrow do(\mathbf{a}) \{\varphi'\}$ .

**Proof:** We need to prove that  $(\varphi[s_i] \wedge (\psi \rightarrow do(\mathbf{a})) = b_i \in s) \Rightarrow \varphi'[s_{i+1}]$ . Therefore, assume  $\varphi[s_i] \wedge (\psi \rightarrow do(\mathbf{a})) = b_i \in s$ . Two cases need to be distinguished: The case that the condition  $\psi$  holds in  $s_i$  and the case that it does not hold in  $s_i$ . In the former case, because we have  $\{\varphi \wedge \psi\} \mathbf{a} \{\varphi'\}$  we then know that  $s_{i+1} \models \varphi'$ . In the latter case, the conditional action is not executed and  $s_{i+1} = s_i$ . From  $((\varphi \wedge \neg\psi) \rightarrow \varphi')[s_i]$ ,  $\varphi[s_i]$  and  $\neg\psi[s_i]$  it then follows that  $\varphi'[s_{i+1}]$  since  $\varphi'$  is a state formula.

The definition of Hoare triples presented here formalises a *total correctness property*. A Hoare triple  $\{\varphi\} b \{\psi\}$  ensures that if initially  $\varphi$  holds, then an attempt to execute  $b$  results in a successor state and in that state  $\psi$  holds. This is different from *partial correctness* where no claims about the termination of actions and the existence of successor states are made.

## 4.2 Basic Action Theories

A *basic action theory* specifies the effects of the basic capabilities of an agent. It specifies when an action is enabled, it specifies the effects of an action and what does not change when an action is executed. Therefore, a basic action theory consists of axioms for the predicate *enabled* for each basic capability, Hoare triples that specify the effects of basic capabilities and Hoare triples that specify frame axioms associated with these capabilities. Since the belief update capabilities of an agent are not fixed by the language GOAL but are user-defined, the user should specify the axioms and Hoare triples for belief update capabilities. The special actions for goal updating **adopt** and **drop** are part of GOAL and a set of axioms and Hoare triples for these actions is specified below.

### 4.2.1 Actions on beliefs: capabilities of the shopping assistant

Because in this paper, our concern is not with the specification of basic action theories in particular, but with providing a programming framework for agents in which such specifications can be plugged in, we only provide some example specifications of the capabilities defined in the personal assistant example that we need in the proof of correctness below.

First, we specify a set of axioms for each of our basic actions that state when that action is enabled. Below, we abbreviate the book titles of the example, and write  $T$  for *The Intentional Stance* and  $I$  for *Intentions, Plans, and Practical*

*Reason.* In the shopping agent example, we then have:

$$\begin{aligned}
& \text{enabled}(\text{goto\_website}(\text{site})) \leftrightarrow \text{true}, \\
& \text{enabled}(\text{search}(\text{book})) \leftrightarrow \text{B}(\text{current\_website}(\text{Amazon.com})), \\
& \text{enabled}(\text{put\_in\_shopping\_cart}(\text{book})) \leftrightarrow \text{B}(\text{current\_website}(\text{book})), \\
& \text{enabled}(\text{pay\_cart}) \leftrightarrow \\
& ((\text{Bin\_cart}(T) \vee \text{Bin\_cart}(I)) \wedge \text{Bcurrent\_website}(\text{ContentCart})).
\end{aligned}$$

Second, we list a number of effect axioms that specify the effects of a capability in particular situations defined by the preconditions of the Hoare triple.

- The action  $\text{goto\_website}(\text{site})$  results in moving to the relevant web page:  
 $\{\text{true}\} \text{goto\_website}(\text{site}) \{\text{Bcurrent\_website}(\text{site})\},$
- At Amazon.com, searching for a book results in finding a page with relevant information about the book:  
 $\{\text{Bcurrent\_website}(\text{Amazon.com})\} \text{search}(\text{book}) \{\text{Bcurrent\_website}(\text{book})\}$
- On the page with information about a particular book, selecting the action  $\text{put\_in\_shopping\_cart}(\text{book})$  results in the book being put in the cart; also, a new web page appears on which the contents of the cart are listed:  
 $\{\text{Bcurrent\_website}(\text{book})\}$   
 $\text{put\_in\_shopping\_cart}(\text{book})$   
 $\{\text{B}(\text{in\_cart}(\text{book}) \wedge \text{current\_website}(\text{ContentCart}))\}$
- In case  $\text{book}$  is in the cart, and the current web page presents a list of all the books in the cart, the action  $\text{pay\_cart}$  may be selected resulting in the buying of all listed books:  
 $\{\text{B}(\text{in\_cart}(\text{book}) \wedge \text{current\_website}(\text{ContentCart}))\}$   
 $\text{pay\_cart}$   
 $\{\neg \text{Bin\_cart}(\text{book}) \wedge \text{B}(\text{bought}(\text{book}) \wedge \text{current\_website}(\text{Amazon.com}))\}$

Finally, we need a number of frame axioms that specify which properties are not changed by each of the capabilities of the agent. For example, both the capabilities  $\text{goto\_website}(\text{site})$  and  $\text{search}(\text{book})$  do not change any beliefs about  $\text{in\_cart}$ . Thus we have, e.g.:

$$\begin{aligned}
& \{\text{Bin\_cart}(\text{book})\} \text{goto\_website}(\text{site}) \{\text{Bin\_cart}(\text{book})\} \\
& \{\text{Bin\_cart}(\text{book})\} \text{search}(\text{book}) \{\text{Bin\_cart}(\text{book})\}
\end{aligned}$$

It will be clear that we need more frame axioms than these two, and some of these will be specified below in the proof of the correctness of the shopping agent.

It is important to realise that the only Hoare triples that need to be specified for agent capabilities are Hoare triples that concern the effects upon the *beliefs* of the agent. Changes and persistence of (some) goals due to executing actions can be derived with the proof rules and axioms below that are specifically designed to reason about the effects of actions on goals.

### 4.2.2 Actions on goals

A theory of the belief update capabilities and their effects on the beliefs of an agent must be complemented with a theory about the effects of actions upon the goals of an agent. Such a theory should capture both the effects of the default commitment strategy as well as give a formal specification of the the **drop** and **adopt** actions. Only in Section 4.3 we aim at providing a complete system, in the discussion in the current section, there are dependencies between the axioms and rules discussed.

**Default commitment strategy** The default commitment strategy imposes a constraint on the persistence of goals. A goal persists if it is not the case that after doing  $a$  the goal is believed to be achieved. Only action  $\text{drop}(\phi)$  is allowed to overrule this constraint. Therefore, in case  $a \neq \text{drop}(\phi)$ , we have that  $\{G\phi\} a \{B\phi \vee G\phi\}$  (using the rule for conditional actions from Table 9, one can derive that this triple also holds for general conditional actions  $b$ , rather than just actions  $a$ ). The Hoare triple precisely captures the default commitment strategy and states that after executing an action the agent either believes it has achieved  $\phi$  or it still has the goal  $\phi$  if  $\phi$  was a goal initially.

$\frac{a \neq \text{drop}(\phi)}{\{G\phi\} a \{B\phi \vee G\phi\}}$
---

Table 4: Persistence of goals

A similar Hoare triple can be given for the persistence of the absence of a goal. Formally, we have

$$\{\neg G\phi\} b \{\neg B\phi \vee \neg G\phi\} \quad (1)$$

This Hoare triple states that the absence of a goal  $\phi$  persists, and in case it does not persist the agent does not believe  $\phi$  (anymore). The adoption of a goal may be the result of executing an **adopt** action, of course. However, it may also be the case that an agent believed it achieved  $\phi$  but after doing  $b$  no longer believes this to be the case and adopts  $\phi$  as a goal again. For example, if the goal base is  $\{p \wedge q\}$  and the belief base  $\Sigma = \{p\}$ , then the agent does not have a goal to achieve  $p$  because it already believes  $p$  to be the case; however, in case an action changes the belief base such that  $p$  is no longer is believed, the agent has a goal to achieve  $p$  (again). This provides for a mechanism similar to that of maintenance goals. We do not need the Hoare triple (1) as an axiom, however, since it is a direct consequence of the fact that  $B\phi \rightarrow \neg G\phi$  (this is exactly the postcondition of (1). Note that the stronger  $\{\neg G\phi\} b \{\neg G\phi\}$  does not hold, even if  $b \neq \varphi \rightarrow \text{do}(\text{adopt}(\phi))$ . This occurs for example if we have  $G(p \wedge q) \wedge Bp$ . Then the agent does not have  $p$  as a goal, since he believes it has already been achieved, but, if he would give up  $p$  as a belief, it becomes to be a goal.

In the semantics of Hoare triples (Definition 4.2) we stipulated that if  $\mathbf{a}$  is not enabled, we verify the postcondition in the same state as the pre-condition:

$\frac{\varphi \rightarrow \neg \text{enabled}(\mathbf{a})}{\{\varphi\} \mathbf{a} \{\varphi\}}$
--

Table 5: Infeasible actions

**Frame properties on Beliefs** The specification of the special actions **drop** and **adopt** involves a number of frame axioms and a number of proof rules. The frame axioms capture the fact that neither of these actions has any effect on the beliefs of an agent. Note that, combining such properties with e.g. the Consequence Rule (Table 10) one can derive the triple  $\{\mathbf{B}\psi\} \text{adopt}(\phi) \{\neg \mathbf{G}\psi\}$ .

$\begin{array}{ll} \{\mathbf{B}\phi\} \text{adopt}(\psi) \{\mathbf{B}\phi\} & \{\neg \mathbf{B}\phi\} \text{adopt}(\psi) \{\neg \mathbf{B}\phi\} \\ \{\mathbf{B}\phi\} \text{drop}(\psi) \{\mathbf{B}\phi\} & \{\neg \mathbf{B}\phi\} \text{drop}(\psi) \{\neg \mathbf{B}\phi\} \end{array}$
--

Table 6: Frame Properties on Beliefs for **adopt** and **drop**

**(Non-)effects of adopt** The proof rules for the actions **adopt** and **drop** capture the effects on the goals of an agent. For each action, we list proof rules for the effect and the persistence (‘non-effect’) on the goal base for adoption (Table 7) and dropping (Table 8) of goals, respectively.

An agent adopts a new goal  $\phi$  in case the agent does not believe  $\phi$  and  $\phi$  is not a contradiction. Concerning persistence, an adopt action does not remove any current goals of the agent. Any existing goals thus persist when **adopt** is executed. The persistence of the absence of goals is somewhat more complicated in the case of an **adopt** action. An **adopt**( $\phi$ ) action does not add a new goal  $\psi$  in case  $\psi$  is not entailed by  $\phi$  or  $\psi$  is believed to be the case:

Effects of adopt
$\frac{\not\models_C \neg \phi}{\{\neg \mathbf{B}\phi\} \text{adopt}(\phi) \{\mathbf{G}\phi\}}$
Non-effect of adopt
$\frac{\not\models_C \psi \rightarrow \phi}{\{\mathbf{G}\psi\} \text{adopt}(\phi) \{\mathbf{G}\psi\} \quad \{\neg \mathbf{G}\phi\} \text{adopt}(\psi) \{\neg \mathbf{G}\phi\}}$

Table 7: (Non-)effects of **adopt**

A drop action **drop**( $\phi$ ) results in the removal of all goals that entail  $\phi$ . This is captured by the first proof rule in Table 8

Effects of drop	$\frac{\models_C \psi \rightarrow \phi}{\{G\psi\} \text{drop}(\phi) \{\neg G\psi\}}$
Non-Effects of drop	$\{ \neg G\phi \} \text{drop}(\psi) \{ \neg G\phi \} \quad \{ \neg G(\phi \wedge \psi) \wedge G\phi \} \text{drop}(\psi) \{ G\phi \}$

Table 8: (Non-)effects of drop

Concerning persistence of goals under **drop**: a drop action  $\text{drop}(\phi)$  never results in the adoption of new goals. The absence of a goal  $\psi$  thus persists when a drop action is executed. It is more difficult to formalise the persistence of a goal with respect to a drop action. Since a drop action  $\text{drop}(\phi)$  removes goals which entail  $\phi$ , to conclude that a goal  $\psi$  persists after executing the action, we must make sure that the goal does not depend on a goal (is a subgoal) that is removed by the drop action. In case the conjunction  $\phi \wedge \psi$  is not a goal, we know this for certain.

The basic action theories for GOAL include a number of proof rules to derive Hoare triples. The Rule for Infeasible Actions (Table 5) allows to derive frame axioms for an action in case it is not enabled in a particular situation. The Rule for Conditional Actions allows the derivation of Hoare triples for conditional actions from Hoare triples for capabilities. This rule is justified by lemma 4.3. Finally, there are three rules for combining Hoare triples and for strengthening the precondition and weakening the postcondition: they are displayed in Table 10.

Finally, we list how one goes from simple actions  $a$  to conditional actions  $b$ , and how to use complex formulas in pre- and post-conditions.

$\frac{\{ \varphi \wedge \psi \} a \{ \varphi' \}, (\varphi \wedge \neg \psi) \rightarrow \varphi'}{\{ \varphi \} \psi \rightarrow do(a) \{ \varphi' \}}$
---

Table 9: Rule for conditional actions

We did not aim in this section at giving a weakest set of rules: in fact, the rules that manipulate the pre- and post-conditions in Table 10 for conditional actions  $b$  could already be derived if we had only given them from simple actions  $a$ .

### 4.3 A Complete Hoare System

We now address the issue of finding a complete Hoare system for GOAL. Let  $\vdash_H \{ \rho \} S \{ \sigma \}$  denote that the Hoare triple with pre-condition  $\rho$  and postcondition  $\sigma$  is derivable in the calculus  $H$  that we are about to introduce, and let  $\models_H$  denote the truth of such assertions, that is,  $\models_H$  determines the truth of mental state

<p>Consequence Rule:</p> $\frac{\varphi' \rightarrow \varphi, \{\varphi\} \mathbf{a} \{\psi\}, \psi \rightarrow \psi'}{\{\varphi'\} \mathbf{a} \{\psi'\}}$	<p>Conjunction Rule:</p> $\frac{\{\varphi_1\} b \{\psi_1\}, \{\varphi_2\} b \{\psi_2\}}{\{\varphi_1 \wedge \varphi_2\} b \{\psi_1 \wedge \psi_2\}}$
<p>Disjunction Rule:</p> $\frac{\{\varphi_1\} b \{\psi\}, \{\varphi_2\} b \{\psi\}}{\{\varphi_1 \vee \varphi_2\} b \{\psi\}}$	

Table 10: Structural rules

formulas (Definition 2.3), that of formulas with *enabled*( $-$ ) (Definition 2.12) and that of Hoare triples (Definition 4.1), on mental states  $\langle \Sigma, \Gamma \rangle$ . From now on, the statement  $S$  ranges over basic actions  $\mathbf{a}$  and conditional actions  $b$ . Then, this subsection wants to settle whether our calculus  $H$  is sound and complete, i.e. whether it can be proven that, for any pre- and postcondition  $\rho$  and  $\sigma \in \mathcal{L}_M$ ,

$$\vdash_H \{\rho\} S \{\sigma\} \iff \models_H \{\rho\} S \{\sigma\}$$

Finding such a complete system is, even for ‘ordinary deterministic programs’, impossible, since such programs are interpreted over domains that include the integers, and by Gödel’s Incompleteness Theorem, we know that axiomatize a domain that includes those integers (cf. [3]). Here, our domain is not that of the integers, but, instead, we will assume a completeness result for the basic capabilities  $Bcap$  that modify the belief base, so that we can concentrate on the actions that modify the goals.

**Definition 4.4** (*General Substitutions*)

We define the following general substitution scheme. Let  $\sigma, \alpha, \beta$  be mental state formulas, and  $X$  a variable ranging over formulas from  $\mathcal{L}$ . Then  $\beta(X)$  denotes that  $X$  may occur in  $\beta$ . Let  $C(X)$  denote a condition on  $X$ . Then  $\sigma[\alpha/\beta(X) \mid C(X)]$  denotes the formula that is obtained from  $\sigma$  by substituting all occurrences in  $\sigma$  of  $\beta(X)$  for which  $C(X)$  holds, by  $\alpha$ .

For instance, the result of  $(Gp \wedge \neg Gq \wedge Gs)[Br/G\phi \mid (p \wedge q) \rightarrow \phi]$  is  $Br \wedge \neg Br \wedge Gs$ .

**Definition 4.5** (The system H)

The valid Hoare triples for GOAL are as follows:

<i>Belief Capabilities</i>	$\{\rho(\mathbf{a})\} \mathbf{a} \{\sigma(\mathbf{a})\}$
<i>Adopt</i>	$\{(enabled(\mathbf{adopt}(\phi)) \wedge \sigma[\neg \mathbf{B}\phi' / \mathbf{G}\phi' \mid \vdash_C \phi \rightarrow \phi']) \vee (\neg enabled(\mathbf{adopt}(\phi)) \wedge \sigma)\} \{\mathbf{adopt}(\phi)\} \{\sigma\}$
<i>Drop</i>	$\{\sigma[\mathbf{true} / \neg \mathbf{G}\phi' \mid \vdash_C \phi' \rightarrow \phi]\} \mathbf{drop}(\phi) \{\sigma\}$
<i>Conditional Actions</i>	$\frac{\{\rho \wedge \psi\} \mathbf{a} \{\sigma\}, \models_{ME} (\sigma \wedge \neg \psi) \rightarrow \rho}{\{\rho\} \psi \rightarrow do(\mathbf{a}) \{\sigma\}}$
<i>Consequence Rule</i>	$\frac{\models_{ME} \sigma \rightarrow \varphi, \{\varphi\} \mathbf{a} \{\psi\}, \models_{ME} \psi \rightarrow \sigma}{\{\rho\} \mathbf{a} \{\sigma\}}$

**Lemma 4.6** (Substitution Lemma)

- (i) Let  $\langle \Sigma, \Gamma \rangle \models_M enabled(\mathbf{adopt}(\phi))$  and let  $\langle \Sigma', \Gamma' \rangle = \mathcal{M}(\mathbf{adopt}(\phi), \langle \Sigma, \Gamma \rangle)$ . Then:

$$\langle \Sigma, \Gamma \rangle \models_M \sigma[\neg \mathbf{B}\phi' / \mathbf{G}\phi' \mid \vdash_C \phi \rightarrow \phi'] \iff \langle \Sigma', \Gamma' \rangle \models_M \sigma$$

- (ii) Let  $\langle \Sigma', \Gamma' \rangle = \mathcal{M}(\mathbf{drop}(\phi), \langle \Sigma, \Gamma \rangle)$ . Then:

$$\langle \Sigma, \Gamma \rangle \models_M \sigma[\mathbf{true} / \neg \mathbf{G}\phi' \mid \vdash_C \phi' \rightarrow \phi] \iff \langle \Sigma', \Gamma' \rangle \models_M \sigma$$

**Proof.** We only prove (i), the proof of (ii) is similar. We prove (i) by induction on the mental state formula  $\sigma$ .

1.  $\sigma$  is of the form  $\mathbf{G}\chi$ . We distinguish two cases.
  - (a)  $\vdash_C \phi \rightarrow \chi$ . By definition of  $\mathcal{M}(\mathbf{adopt}(\phi), \langle \Sigma, \Gamma \rangle)$ , we immediately see that  $\langle \Sigma, \Gamma \rangle \models_M \neg \mathbf{B}\chi$  iff  $\langle \Sigma', \Gamma' \rangle \models_M \mathbf{G}\chi$ .
  - (b)  $\not\vdash_C \phi \rightarrow \chi$ . The definition of  $\mathcal{M}(\mathbf{adopt}(\phi), \langle \Sigma, \Gamma \rangle)$  guarantees that the adopt has no effect on the fact that  $\chi$  is a goal, thus we have  $\langle \Sigma, \Gamma \rangle \models_H \mathbf{G}\chi$  iff  $\mathcal{M}(\mathbf{adopt}(\phi), \langle \Sigma, \Gamma \rangle) \models_H \mathbf{G}\chi$ . Also, the substitution has no change as an effect:  $(\mathbf{G}\chi)[\neg \mathbf{B}\phi' / \mathbf{G}\phi' \mid \vdash_C \chi \rightarrow \phi'] = \mathbf{G}\chi$ , and we have the desired result.
2.  $\sigma$  is of the form  $\mathbf{B}\chi$ . In this case, the substitution had no effect, and also, since the **adopt** has no effect on the belief base, we have that  $\langle \Sigma, \Gamma \rangle \models_M \mathbf{B}\chi$  iff  $\langle \Sigma', \Gamma' \rangle \models_M \mathbf{B}\chi$ .
3. The cases that  $\sigma$  is a negation or a conjunction of mental states, follows immediately.

**Lemma 4.7** (Soundness of  $\vdash_H$ )

for any pre- and postcondition  $\rho$  and  $\sigma \in \mathcal{L}_M$ ,  $\vdash_H \{\rho\} S \{\sigma\} \implies \models_H \{\rho\} S \{\sigma\}$

**Proof.** Soundness of *Belief Capabilities* is assumed. The cases for **adopt** and **drop** immediately follow from Lemma 4.6. The soundness of *Consequence Rule* is easily proven using Theorem 2.5. Let us finally consider the rule *Conditional Actions*. Suppose that  $\vdash_H \{\sigma \wedge \psi\} \mathbf{a} \{\rho\}$  (1), and  $\models_{ME} (\rho \wedge \neg\psi) \rightarrow \sigma$  (2), and take an arbitrary mental state  $\langle \Sigma, \Gamma \rangle$ . We have to demonstrate that  $\langle \Sigma, \Gamma \rangle \models_{ME} \{\rho\} \psi \rightarrow do(\mathbf{a}) \{\sigma\}$  (3). Hence, we assume that  $\langle \Sigma, \Gamma \rangle \models_{ME} \rho$ . We then distinguish two cases. First, assume  $\langle \Sigma, \Gamma \rangle \models_{ER} \psi$ . Then, by our assumption (1), we have that  $\langle \Sigma', \Gamma' \rangle \models_M \sigma$ , for  $\langle \Sigma', \Gamma' \rangle = \mathcal{M}(\mathbf{a}, \langle \Sigma, \Gamma \rangle)$ . The second case is the one in which  $\langle \Sigma, \Gamma \rangle \not\models_M \psi$ , i.e.,  $\langle \Sigma, \Gamma \rangle \models_M \neg\psi$ . By (2), we know that  $\langle \Sigma', \Gamma' \rangle \models_{ER} \sigma$ , for any  $\langle \Sigma', \Gamma' \rangle$  for which  $\langle \Sigma, \Gamma \rangle \xrightarrow{b} \langle \Sigma', \Gamma' \rangle$ . All in all, we have proven (3).

We first introduce the notion of weakest liberal precondition, originally due to Dijkstra ([7]). However, we introduce it immediately in the syntax.

**Definition 4.8** (Weakest Liberal Precondition)

For  $S = \mathbf{a}', \mathbf{adopt}(\phi) \mathbf{drop}(\phi)$  and  $\psi \rightarrow \mathbf{a}$ , ( $\mathbf{a}'$  a belief capability,  $\mathbf{a}$  any basic capability), we define the weakest liberal precondition for  $S$  to achieve  $\sigma$ ,  $wlp(S, \sigma) \in \mathcal{L}_m$ , as follows:

1. If  $\mathbf{a}'$  is a belief capability, and  $\vdash_H \{\sigma^{-1}(\mathbf{a}')\} \mathbf{a}' \{\sigma(\mathbf{a}')\}$  is the rule for  $\mathbf{a}'$ , then  $wlp(\mathbf{a}', \sigma(\mathbf{a}')) = \sigma^{-1}$
2.  $wlp(\mathbf{adopt}(\phi), \sigma) = (enabled(\mathbf{adopt}(\phi)) \wedge \sigma[\neg B\phi' / G\phi' \mid \vdash_C \phi \rightarrow \phi']) \vee (\neg enabled(\mathbf{adopt}(\phi)) \wedge \sigma)$
3.  $wlp(\mathbf{drop}(\phi), \sigma) = \sigma[\mathbf{true} / \neg G\phi' \mid \vdash_C \phi' \rightarrow \phi]$
4.  $wlp(\psi \rightarrow \mathbf{a}, \sigma) = (\psi \wedge wlp(\mathbf{a}, \sigma)) \vee (\neg\psi \wedge \sigma)$

Note that the weakest precondition  $wlp(S, \sigma)$  is indeed an mental state formula.

**Lemma 4.9** (Weakest Precondition Lemma)

We have:  $\vdash_H \{wlp(S, \sigma)\} S \{\sigma\}$ , for every  $S$  and every postcondition  $\sigma$ .

**Proof.** For  $S = \mathbf{a}', \mathbf{adopt}(\phi), \mathbf{drop}(\phi)$ , this follows immediately from Definition 4.5. For conditional actions  $b = \psi \rightarrow \mathbf{a}$ , we have to prove

$$\vdash_H \{(\psi \wedge wlp(\mathbf{a}, \sigma)) \vee (\neg\psi \wedge \sigma)\} \psi \rightarrow \mathbf{a} \{\sigma\}$$

Let us abbreviate  $(\psi \wedge wlp(\mathbf{a}, \sigma)) \vee (\neg\psi \wedge \sigma)$  to  $\sigma_b^{-1}$ . The induction hypothesis tells us that  $\vdash_H \{wlp(\mathbf{a}, \sigma)\} \mathbf{a} \{\sigma\}$  (1). Note that  $\models_M \sigma_b^{-1} \rightarrow wlp(\mathbf{a}, \sigma)$ . Hence, by the *Consequence Rule* and (1), we have  $\vdash_H \{\sigma_b^{-1}\} \mathbf{a} \{\sigma\}$  (2). Obviously, we also have  $\models_M (\sigma_b^{-1} \wedge \neg\psi) \rightarrow \sigma$  (3). Applying the *Conditional Actions* rule to (2) and (3), we conclude  $\vdash_H \{\sigma_b^{-1}\} \psi \rightarrow \mathbf{a} \{\sigma\}$ , which was to be proven.

**Lemma 4.10**

$$\vdash_H \{\rho\} S \{\sigma\} \Rightarrow \models_M \rho \rightarrow wlp(S, \sigma)$$



**Proof.** We even prove a stronger statement, i.e., that for all mental states  $\langle \Sigma, \Gamma \rangle$ , if  $\langle \Sigma, \Gamma \rangle \models_H \{\rho\} S \{\sigma\}$ , then  $\langle \Sigma, \Gamma \rangle \models_M \rho \rightarrow wlp(S, \sigma)$ . To prove this, we take an arbitrary  $\langle \Sigma, \Gamma \rangle$  for which both  $\langle \Sigma, \Gamma \rangle \models_M \rho$  and  $\langle \Sigma, \Gamma \rangle \models_H \{\rho\} S \{\sigma\}$  and we then have to show that  $\langle \Sigma, \Gamma \rangle \models_M wlp(S, \sigma)$ .

1.  $S = \text{adopt}(\phi)$ . We know that  $\langle \Sigma, \Gamma \rangle \models_H \{\rho\} \text{adopt}(\phi) \{\sigma\}$ . We distinguish two cases, the first of which says that  $\langle \Sigma, \Gamma \rangle \models_M \neg \text{enabled}(\text{adopt}(\phi))$ . Then we have a transition from  $\langle \Sigma, \Gamma \rangle$  to itself, and hence  $\langle \Sigma, \Gamma \rangle \models_M \sigma$ , and hence  $\langle \Sigma, \Gamma \rangle \models_M \neg \text{enabled}(\text{adopt}(\phi)) \wedge \sigma$ , so that  $\langle \Sigma, \Gamma \rangle \models_M wlp(\text{adopt}(\phi), \sigma)$ . In the second case,  $\langle \Sigma, \Gamma \rangle \models_M \text{enabled}(\text{adopt}(\phi))$ . By the Substitution Lemma 4.6, case (i), we then immediately see  $\langle \Sigma, \Gamma \rangle \models_M \sigma[\neg B\phi' / G\phi' \mid \vdash_C \phi \rightarrow \phi']$ , and hence  $\langle \Sigma, \Gamma \rangle \models_M wlp(\text{adopt}(\phi), \sigma)$ .
2.  $S = \text{drop}(\phi)$ . This case follows immediately from the Substitution Lemma and the definition of  $wlp(\text{drop}(\phi), \sigma)$ .
3.  $S = \psi \rightarrow a$ . We know that  $\langle \Sigma, \Gamma \rangle \models_H \{\rho\} \psi \rightarrow a \{\sigma\}$  and that  $\langle \Sigma, \Gamma \rangle \models_M \rho$ . If  $\langle \Sigma, \Gamma \rangle \models_M \neg \psi$ , then the transition belonging to  $S$  ends up in  $\langle \Sigma, \Gamma \rangle$ , and hence we then have  $\langle \Sigma, \Gamma \rangle \models_M \neg \psi \wedge \sigma$ , and in particular  $\langle \Sigma, \Gamma \rangle \models_M wlp(S, \sigma)$ . In the other case we have  $\langle \Sigma, \Gamma \rangle \models_M \psi$ , and we then know that  $\langle \Sigma, \Gamma \rangle \models_H \{\rho\} a \{\sigma\}$ . By induction we conclude that  $\langle \Sigma, \Gamma \rangle \models_M wlp(a, \sigma)$ , and hence  $\langle \Sigma, \Gamma \rangle \models_M wlp(\psi \rightarrow a, \sigma)$ .

**Theorem 4.11** (Completeness of  $\vdash_H$ )

For any pre- and postcondition  $\rho$  and  $\sigma \in \mathcal{L}_M$ ,

$$\vdash_H \{\rho\} S \{\sigma\} \iff \models_H \{\rho\} S \{\sigma\}$$

**Proof.** Suppose  $\models_M \{\rho\} S \{\sigma\}$ . The Weakest Precondition Lemma (4.9) tells us that  $\vdash_H \{wlp(S, \sigma)\} S \{\sigma\}$  (1). By Lemma 4.10, we have  $\models_M \sigma \rightarrow wlp(S, \sigma)$  (2). We finally apply the *Consequence Rule* to (1) and (2) to conclude to conclude that  $\vdash_H \{\rho\} S \{\sigma\}$ .

## 4.4 Temporal logic

On top of the Hoare triples for specifying actions, a temporal logic is used to specify and verify properties of GOAL agents. Two new operators are introduced. The proposition **init** states that the agent is at the beginning of execution and nothing has happened yet. The second operator **until** is a weak until operator.  $\varphi$  **until**  $\psi$  means that  $\psi$  eventually becomes true and  $\varphi$  is true until  $\psi$  becomes true, or  $\psi$  never becomes true and  $\varphi$  remains true forever.

**Definition 4.12** (language of temporal logic  $\mathcal{L}_T$  based on  $\mathcal{L}$ )

The temporal logic language  $\mathcal{L}_T$  is inductively defined by:

- **init**  $\in \mathcal{L}_T$ ,
- $\text{enabled}(a), \text{enabled}(\varphi \rightarrow \text{do}(a)) \in \mathcal{L}_T$  for  $a \in \text{Cap}$ ,

- if  $\phi \in \mathcal{L}$ , then  $B\phi, G\phi \in \mathcal{L}_T$ ,
- if  $\varphi, \psi \in \mathcal{L}_T$ , then  $\neg\varphi, \varphi \wedge \psi \in \mathcal{L}_T$ ,
- if  $\varphi, \psi \in \mathcal{L}_T$ , then  $\varphi \textbf{ until } \psi \in \mathcal{L}_T$ .

A number of other well known temporal operators can be defined in terms of the operator **until**. The *always* operator  $\Box\varphi$  is an abbreviation for  $\varphi \textbf{ until false}$ , and the *eventuality* operator  $\Diamond\varphi$  is defined as  $\neg\Box\neg\varphi$  as usual.

Temporal formulas are evaluated with respect to a trace  $s$  and a time point  $i$ . State formulas like  $B\phi$ ,  $G\psi$ , *enabled(a)* etc. are evaluated with respect to mental states.

**Definition 4.13** (*semantics of temporal formulas*)

Let  $s$  be a trace and  $i$  be a natural number.

- $s, i \models \textbf{init}$  iff  $i = 0$ ,
- $s, i \models \textit{enabled}(\mathbf{a})$  iff  $\textit{enabled}(\mathbf{a})[s_i]$ ,
- $s, i \models \textit{enabled}(\varphi \rightarrow \textit{do}(\mathbf{a}))$  iff  $\textit{enabled}(\varphi \rightarrow \textit{do}(\mathbf{a}))[s_i]$ ,
- $s, i \models B\phi$  iff  $B\phi[s_i]$ ,
- $s, i \models G\phi$  iff  $G\phi[s_i]$ ,
- $s, i \models \neg\varphi$  iff  $s, i \not\models \varphi$ ,
- $s, i \models \varphi \wedge \psi$  iff  $s, i \models \varphi$  and  $s, i \models \psi$ ,
- $s, i \models \varphi \textbf{ until } \psi$  iff  $\exists j \geq i (s, j \models \psi \wedge \forall k (i \leq k < j (s, k \models \varphi)))$  or  $\forall k \geq i (s, k \models \varphi)$ .

We are particularly interested in temporal formulas that are valid with respect to the set of traces  $S_A$  associated with a GOAL agent  $A$ . Temporal formulas valid with respect to  $S_A$  express properties of the agent  $A$ .

**Definition 4.14** Let  $S$  be a set of traces.

- $S \models \varphi$  iff  $\forall s \in S, i(s, i \models \varphi)$ ,
- $\models \varphi$  iff  $S \models \varphi$  where  $S$  is the set of all traces.

In general, two important types of temporal properties can be distinguished. Temporal properties are divided into *liveness* and *safety* properties. Liveness properties concern the progress that a program makes and express that a (good) state eventually will be reached. Safety properties, on the other hand, express that some (bad) state will never be entered. In the rest of this section, we discuss a number of specific liveness and safety properties of an agent  $A = \langle \Pi_A, \Sigma_0, \Gamma_0 \rangle$ .

We show that each of the properties that we discuss are equivalent to a set of Hoare triples. The importance of this result is that it shows that temporal

properties of agents can be proven by inspection of the program text only. The fact that proofs of agent properties can be constructed by inspection of the program text means that there is no need to reason about individual traces of an agent or its operational behaviour. In general, reasoning about the program text is more economical since the number of traces associated with a program is exponential in the size of the program.

The first property we discuss concerns a safety property, and is expressed by the temporal formula  $\varphi \rightarrow (\varphi \textbf{ until } \psi)$ . Properties in this context always refer to agent properties and are evaluated with respect to the set of traces associated with that agent. Therefore, we can explain the informal meaning of the property as stating that if  $\varphi$  ever becomes true, then it remains true until  $\psi$  becomes true. By definition, we write this property as  $\varphi \textbf{ unless } \psi$ :

$$\varphi \textbf{ unless } \psi \stackrel{df}{=} \varphi \rightarrow (\varphi \textbf{ until } \psi)$$

An important special case of an **unless** property is  $\varphi \textbf{ unless false}$ , which expresses that if  $\varphi$  ever becomes true, it will remain true.  $\varphi \textbf{ unless false}$  means that  $\varphi$  is a *stable* property of the agent. In case we also have  $\textbf{init} \rightarrow \varphi$ , where **init** denotes the initial starting point of execution,  $\varphi$  is always true and is an *invariant* of the program.

Now we show that **unless** properties of an agent  $A = \langle \Pi, \sigma_0, \gamma_0 \rangle$  are equivalent to a set of Hoare triples for basic actions in  $\Pi$ . This shows that we can prove **unless** properties by proving a finite set of Hoare triples. The proof relies on the fact that if we can prove that after executing any action from  $\Pi$  either  $\varphi$  persists or  $\psi$  becomes true, we can conclude that  $\varphi \textbf{ unless } \psi$ .

**Theorem 4.15** Let  $A = \langle \Pi_A, \Sigma_0, \Gamma_0 \rangle$ . Then:

$$\forall b \in \Pi_A (\{\varphi \wedge \neg\psi\} b \{\varphi \vee \psi\}) \text{ iff } S_A \models \varphi \textbf{ unless } \psi$$

**Proof:** The proof from right to left is the easiest direction in the proof. Suppose  $S_A \models \varphi \textbf{ unless } \psi$  and  $s, i \models \varphi$ . This implies that  $s, i \models \varphi \textbf{ until } \psi$ . In case we also have  $s, i \models \psi$ , we are done. So, assume  $s, i \models \neg\psi$  and action  $b$  is selected in the trace at state  $s_i$ . From the semantics of **until** we then know that  $\varphi \vee \psi$  holds at state  $s_{i+1}$ , and we immediately obtain  $\{\varphi \wedge \neg\psi\} b \{\varphi \vee \psi\}$  since  $s$  and  $i$  were arbitrarily chosen trace and time point. To prove the Hoare triple for the other actions in the agent program  $A$ , note that when we replace action  $b$  with another action  $c$  from  $\Pi_A$  in trace  $s$ , the new trace  $s'$  is still a valid trace that is in the set  $S_A$ . Because we have  $S_A \models \varphi \textbf{ unless } \psi$ , we also have  $s', i \models \varphi \textbf{ unless } \psi$  and from reasoning by analogy we obtain the Hoare triple for action  $c$  (and similarly for all other actions).

We prove the left to right case by contraposition. Suppose that

$$(*) \quad \forall b \in \Pi_A (\{\varphi \wedge \neg\psi\} b \{\varphi \vee \psi\})$$

and for some  $s \in S_A$  we have  $s, i \not\models \varphi \textbf{ unless } \psi$ . The latter fact means that we have  $s, i \models \varphi$  and  $s, i \not\models \varphi \textbf{ until } \psi$ .  $s, i \not\models \varphi \textbf{ until } \psi$  implies that either (i)  $\psi$

is never established at some  $j \geq i$  but we do have  $\neg\varphi$  at some time point  $k > i$  or (ii)  $\psi$  is established at some time  $j > i$ , but in between  $i$  and any such  $j$  it is not always the case that  $\varphi$  holds.

In the first case (i), let  $k > i$  be the smallest  $k$  such that  $s, k \not\models \varphi$ . Then, we have  $s, k-1 \models \varphi \wedge \neg\psi$  and  $s, k \models \neg\varphi \wedge \neg\psi$ . In state  $s_{k-1}$ , however, either a conditional action is performed or no action is performed. From (\*) we then derive a contradiction.

In the second case (ii), let  $k > i$  be the smallest  $k$  such that  $s, k \models \psi$ . Then we know that there is a smallest  $j$  such that  $i < j < k$  and  $s, j \not\models \varphi$  ( $j \neq i$  since  $s, i \models \varphi$ ). This means that we have  $s, j-1 \models \varphi \wedge \neg\psi$ . However, in state  $s_j$  either a conditional action is performed or no action is performed. From (\*) we then again derive a contradiction.

Liveness properties involve eventualities which state that some state will be reached starting from a particular situation. To express a special class of such properties, we introduce the operator  $\varphi$  **ensures**  $\psi$ .  $\varphi$  **ensures**  $\psi$  informally means that  $\varphi$  guarantees the realisation of  $\psi$ , and is defined as:

$$\varphi \text{ ensures } \psi \stackrel{df}{=} \varphi \text{ unless } \psi \wedge (\varphi \rightarrow \diamond\psi)$$

$\varphi$  **ensures**  $\psi$  thus ensures that  $\psi$  is eventually realised starting in a situation in which  $\varphi$  holds, and requires that  $\varphi$  holds until  $\psi$  is realised. For the class of **ensures** properties, we can show that these properties can be proven by proving a set of Hoare triples. The proof of a **ensures** property thus can be reduced to the proof of a set of Hoare triples.

**Theorem 4.16** Let  $A = \langle \Pi_A, \sigma_0, \gamma_0 \rangle$ . Then:

$$\begin{aligned} & \forall b \in \Pi_A(\{\varphi \wedge \neg\psi\} \ b \ \{\varphi \vee \psi\}) \wedge \exists b \in \Pi_A(\{\varphi \wedge \neg\psi\} \ b \ \{\psi\}) \\ & \Rightarrow S_A \models \varphi \text{ ensures } \psi \end{aligned}$$

**Proof:** In the proof, we need the weak fairness assumption. Since  $\varphi$  **ensures**  $\psi$  is defined as  $\varphi$  **unless**  $\psi \wedge (\varphi \rightarrow \diamond\psi)$ , by theorem 4.15 we only need to prove that  $S_A \models \varphi \rightarrow \diamond\psi$  given that  $\forall b \in \Pi_A(\{\varphi \wedge \neg\psi\} \ b \ \{\varphi \vee \psi\}) \wedge \exists b \in \Pi_A(\{\varphi \wedge \neg\psi\} \ b \ \{\psi\})$ . Now suppose, to arrive at a contradiction, that for some time point  $i$  and trace  $s \in S_A$  we have:  $s, i \models \varphi \wedge \neg\psi$  and assume that for all later points  $j > i$  we have  $s, j \models \neg\psi$ . In that case, we know that for all  $j > i$  we have  $s, j \models \varphi \wedge \neg\psi$  (because we may assume  $\varphi$  **unless**  $\psi$ ). However, we also know that there is an action  $b$  that is enabled in a state in which  $\varphi \wedge \neg\psi$  holds and transforms this state to a state in which  $\psi$  holds. The action  $b$  thus is always enabled, but apparently never taken. This is forbidden by weak fairness, and we arrive at a contradiction.

Finally, we introduce a third temporal operator ‘leads to’  $\mapsto$ . The operator  $\varphi \mapsto \psi$  differs from **ensures** in that it does not require  $\varphi$  to remain true until  $\psi$  is established, and is derived from the **ensures** operator.  $\mapsto$  is defined as the transitive, disjunctive closure of **ensures**.

**Definition 4.17** (*leads to operator*)

The leads to operator  $\mapsto$  is defined by:

$\frac{\varphi \text{ ensures } \psi}{\varphi \mapsto \psi}$	$\frac{\varphi \mapsto \chi, \chi \mapsto \psi}{\varphi \mapsto \psi}$	$\frac{\varphi_1 \mapsto \psi, \dots, \varphi_n \mapsto \psi}{(\varphi_1 \vee \dots \vee \varphi_n) \mapsto \psi}$
--	--	--

The meaning of the ‘leads to’ operator is captured by the following lemma.  $\varphi \mapsto \psi$  means that given  $\varphi$  condition  $\psi$  will eventually be realised. The proof of the lemma is an easy induction on the definition of  $\mapsto$ .

**Lemma 4.18**  $\varphi \mapsto \psi \models \varphi \rightarrow \diamond\psi$ .

## 5 Proving Agents Correct

In this section, we use the programming logic to prove the correctness of our example shopping agent. We do not present all the details, but provide enough details to illustrate the use of the programming logic. Before we discuss what it means that an agent program is correct and provide a proof which shows that our example agent is correct, we introduce some notation. The notation involves a number of abbreviations concerning names and propositions in the language of our example agent:

- Instead of *current\_website(sitename)* we just write *sitename*; e.g., we write *Am.com* and *ContentCart* instead of *current\_website(Am.com)* and *current\_website(ContentCart)*, respectively.
- As before, the book titles *The Intentional Stance* and *Intentions, Plans and Practical Reason* that the agent intends to buy are abbreviated to *T* and *I* respectively. These conventions can result in formulas like  $B(T)$ , which means that the agent is at the web page concerning the book *The Intentional Stance*.

A simple and intuitive correctness property, which is natural in this context and is applicable to our example agent, states that a GOAL agent is *correct* when the agent program realises the initial goals of the agent. For this subclass of correctness properties, we may consider the agent to be finished upon establishing the initial goals and in that case the agent could be terminated. Of course, it is also possible to continue the execution of such agents. This class of correctness properties can be expressed by means of temporal formulas like  $G\phi \rightarrow \diamond\neg G\phi$ . Other correctness properties are conceivable, of course, but not all of them can be expressed easily in the temporal proof logic for GOAL.

### 5.1 Correctness Property of the Shopping Agent

From the discussion above, we conclude that the interesting property to prove for our example program is the following property:

$$Bcond \wedge G(bought(T) \wedge bought(I)) \mapsto B(bought(T) \wedge bought(I))$$

where  $Bcond$  is some condition of the initial beliefs of the agent. More specifically,  $Bcond$  is defined by:

$$\begin{aligned} & B_{current\_webpage}(hpage(user)) \wedge \neg Bin\_cart(T) \wedge \neg Bin\_cart(I) \wedge \\ & B(\forall s, s'((s \neq s' \wedge current\_webpage(s)) \rightarrow \neg current\_webpage(s'))) \end{aligned}$$

The correctness property states that the goal to buy the books *The Intentional Stance* and *Intentions, Plans and Practical Reason*, given some initial conditions on the beliefs of the agent, leads to buying (or believing to have bought) these books. Note that this property expresses a *total correctness* property. It states both that the program behaves as desired and that it will eventually reach the desired goal state. An extra reason for considering this property to express correctness of our example agent is that the goals involved once they are achieved remain true forever (they are ‘stable’ properties).

## 5.2 Invariants and Frame Axioms

To be able to prove correctness, we need a number of frame axioms. There is a close relation between frame axioms and invariants of a program. This is because frame axioms express properties that are not changed by actions, and a property that, once true, remains true whatever action is performed is a stable property. In case such a property also holds initially, the property is an *invariant* of the program. In our example program, there is one invariant that states that it is impossible to be at two web pages at the same time:  $inv = B \forall s, s'((s \neq s' \wedge current\_webpage(s)) \rightarrow \neg current\_webpage(s'))$ .

To prove that  $inv$  is an invariant of the agent, we need frame axioms stating that when  $inv$  holds before the execution of an action it still holds after executing that action. Formally, for each  $a \in Cap$ , we need:  $\{inv\} a \{inv\}$ . These frame axioms need to be specified by the user, and for our example agent we assume that they are indeed true. By means of the Consequence Rule (strengthen the precondition of the Hoare triples for capabilities  $a$ ) and the Rule for Conditional Actions (instantiate  $\varphi$  and  $\varphi'$  with  $inv$ ), we then obtain that  $\{inv\} b \{inv\}$  for all  $b \in \Pi$ . By theorem 4.15, we then know that  $inv$  **unless** false. Because we also have that initially  $inv$  holds since  $\langle \sigma_0, \gamma_0 \rangle \models inv$ , we may conclude that **init**  $\rightarrow Bin_v \wedge inv$  **unless** false.  $inv$  thus is an invariant and holds at all times during the execution of the agent. Because of this fact, we do not mention  $inv$  explicitly anymore in the proofs below, but will freely use the property when we need it.

A second property that is stable is the property  $status(book)$ :

$$status(book) \stackrel{df}{=} (Bin\_cart(book) \wedge Gbought(book)) \vee Bbought(book)$$

The fact that  $status(book)$  is stable means that once a book is in the cart and it is a goal to buy the book, it remains in the cart and is only removed from the cart when it is bought.

The proof obligations to prove that  $status(book)$  is a *stable* property, i.e. to prove that  $status(book)$  **unless** false, consist of supplying proofs for

$$\{status(book)\} b \{status(book)\}$$

for each conditional action  $b \in \Pi$  of the shopping agent (cf. theorem 4.15). By the Rule for Conditional Actions, therefore, it is sufficient to prove for each conditional action  $\psi \rightarrow do(a) \in \Pi$  that  $\{status(book) \wedge \psi\} a \{status(book)\}$  and  $(status(book) \wedge \neg\psi) \rightarrow status(book)$ . The latter implication is trivial. Moreover, it is clear that to prove the Hoare triples it is sufficient to prove  $\{status(book)\} a \{status(book)\}$  since we can strengthen the precondition by means of the Consequence Rule. The proof obligations thus reduce to proving  $\{status(book)\} a \{status(book)\}$  for each capability of the shopping agent.

Again, we cannot prove these Hoare triples without a number of frame axioms. Because no capability is allowed to reverse the fact that a book has been bought, for each capability, we can specify a frame axiom for the predicate *bought*:

$$(1) \quad \{Bbought(book)\} a \{Bbought(book)\}$$

In case the book is not yet bought, selecting action *pay\_cart* may change the contents of the cart and therefore we first treat the other three actions *goto\_website*, *search*, and *put\_in\_shopping\_cart* which are not supposed to change the contents of the cart. For each of the latter three capabilities we therefore add the frame axioms:

$$\{Bin\_cart(book) \wedge \neg Bbought(book)\} a \{Bin\_cart(book) \wedge \neg Bbought(book)\}$$

where  $a \neq pay\_cart$ . Note that these frame axioms do not refer to goals but only refer to the beliefs of the agent, in agreement with our claim that only Hoare triples for belief updates need to be specified by the user. By using the axiom  $Gbought(book) \rightarrow \neg Bbought(book)$  and the Consequence Rule, however, we can conclude that:

$$\{Bin\_cart(book) \wedge Gbought(book)\} a \{Bin\_cart(book) \wedge \neg Bbought(book)\}$$

By combining this with the axiom

$$\{Gbought(book)\} a \{Bbought(book) \vee Gbought(book)\}$$

by means of the Conjunction Rule and by rewriting the postcondition with the Consequence Rule, we then obtain

$$(2) \{Bin\_cart(book) \wedge Gbought(book)\} a \{Bin\_cart(book) \wedge Gbought(book)\}$$

where  $a \neq pay\_cart$ . By weakening the postconditions of (1) and (2) by means of the Consequence Rule and combining the result with the Disjunction Rule, it is then possible to conclude that  $\{status(book)\} a \{status(book)\}$  for  $a \neq pay\_cart$ .

As before, in the case of capability *pay\_cart* we deal with each of the disjuncts of *status(book)* in turn. The second disjunct can be handled as before, but the first disjunct is more involved this time because *pay\_cart* can change both the content of the cart and the goal to buy a book if it is enabled. Note that *pay\_cart* only is enabled in case *BContentCart* holds. In case *BContentCart* holds and *pay\_cart* is enabled, from the effect axiom for *pay\_cart* and the Consequence Rule we obtain

$$(3) \quad \frac{\{Bin\_cart(book) \wedge Gbought(book) \wedge BContentCart\}}{pay\_cart} \{Bbought(book)\}$$

In case  $\neg BContentCart$  holds and *pay\_cart* is not enabled, we use the Rule for Infeasible Capabilities to conclude that

$$(4) \quad \frac{\{Bin\_cart(book) \wedge Gbought(book) \wedge \neg BContentCart\}}{pay\_cart} \{Bin\_cart(book) \wedge Gbought(book) \wedge \neg BContentCart\}$$

By means of the Consequence Rule and the Disjunction Rule, we then can conclude from (1), (3) and (4) that  $\{status(book)\} pay\_cart \{status(book)\}$ , and we are done.

### 5.3 Proof Outline

The main proof steps to prove our agent example correct are listed next. The proof steps below consists of a number of **ensures** formulas which together prove that the program reaches its goal in a finite number of steps.

- (1)  $Bhpage(user) \wedge \neg Bin\_cart(T) \wedge Gbought(T) \wedge \neg Bin\_cart(I) \wedge Gbought(I)$   
**ensures**  
 $BAm.com \wedge \neg Bin\_cart(T) \wedge Gbought(T) \wedge \neg Bin\_cart(I) \wedge Gbought(I)$
- (2)  $BAm.com \wedge \neg Bin\_cart(T) \wedge Gbought(T) \wedge \neg Bin\_cart(I) \wedge Gbought(I)$   
**ensures**  
 $[(B(T) \wedge Gbought(T) \wedge \neg Bin\_cart(I) \wedge Gbought(I)) \vee (B(I) \wedge Gbought(I) \wedge \neg Bin\_cart(T) \wedge Gbought(T))]$
- (3)  $B(T) \wedge Gbought(T) \wedge \neg Bin\_cart(I) \wedge Gbought(I)$   
**ensures**  
 $Bin\_cart(T) \wedge Gbought(T) \wedge \neg Bin\_cart(I) \wedge Gbought(I) \wedge BContentCart$
- (4)  $Bin\_cart(T) \wedge Gbought(T) \wedge \neg Bin\_cart(I) \wedge Gbought(I)$   
**ensures**  
 $BAm.com \wedge \neg Bin\_cart(I) \wedge Gbought(I) \wedge status(T)$



- (5)  $B(Am.com) \wedge \neg Bin\_cart(I) \wedge Gbought(I) \wedge status(T)$   
**ensures**  
 $B(I) \wedge Gbought(I) \wedge status(T)$
- (6)  $B(I) \wedge Gbought(I) \wedge status(T)$   
**ensures**  
 $Bin\_cart(I) \wedge Gbought(I) \wedge BContentCart \wedge status(T)$
- (7)  $Bin\_cart(I) \wedge Gbought(I) \wedge BContentCart \wedge status(T)$   
**ensures**  
 $Bbought(T) \wedge Bbought(I)$

At step 3, the proof is split up into two subproofs, one for each of the disjuncts of the disjunct that is ensured in step 2. The proof for the other disjunct is completely analogous. By applying the rules for the ‘leads to’ operator the third to seventh step result in:

- (a)  $B(T) \wedge Gbought(T) \wedge \neg Bin\_cart(I) \wedge Gbought(I) \mapsto$   
 $Bbought(T) \wedge Bbought(I)$
- (b)  $B(I) \wedge Gbought(I) \wedge \neg Bin\_cart(T) \wedge Gbought(T) \mapsto$   
 $Bbought(T) \wedge Bbought(I)$

Combining (a) and (b) by the disjunction rule for the ‘leads to’ operator and by using the transitivity of ‘leads to’ we then obtain the desired correctness result:

$$Bcond \wedge G(bought(T) \wedge bought(I)) \mapsto B(bought(T) \wedge bought(I))$$

with  $Bcond$  as defined previously.

**Step 1** We now discuss the first proof step in somewhat more detail. The remainder of the proof is left to the reader. The proof of a formula  $\varphi$  **ensures**  $\psi$  requires that we show that every action  $b$  in the Personal Assistant program satisfies the Hoare triple  $\{\varphi \wedge \neg\psi\} b \{\varphi \vee \psi\}$  and that there is at least one action  $b'$  which satisfies the Hoare triple  $\{\varphi \wedge \neg\psi\} b' \{\psi\}$ . By inspection of the program, in our case the proof obligations turn out to be:

$$\begin{array}{l} \{Bhpge(user) \wedge \neg Bin\_cart(T) \wedge Gbought(T) \wedge \neg Bin\_cart(I) \wedge Gbought(I)\} \\ \quad b \\ \{Bhpge(user) \wedge \neg Bin\_cart(T) \wedge Gbought(T) \wedge \neg Bin\_cart(I) \wedge Gbought(I)\} \end{array}$$

where  $b$  is one of the actions

$$\begin{array}{l} B(Am.com) \wedge \neg B(in\_cart(book)) \wedge G(bought(book)) \rightarrow do(search(book)), \\ B(book) \wedge G(bought(book)) \rightarrow do(put\_in\_shopping\_cart(book)), \\ B(in\_cart(book)) \wedge G(bought(book)) \rightarrow do(pay\_cart) \end{array}$$

and

$$\begin{aligned}
& \{Bhp\text{age}(user) \wedge \neg Bin\_cart(T) \wedge Gbought(T) \wedge \neg Bin\_cart(I) \wedge Gbought(I)\} \\
& B(hp\text{age}(user) \vee ContentCart) \wedge G(bought(book)) \rightarrow do(goto\_website(Am.com)) \\
& \{BAm.com \wedge \neg Bin\_cart(T) \wedge Gbought(T) \wedge \neg Bin\_cart(I) \wedge Gbought(I)\}
\end{aligned}$$

The proofs of the first three Hoare triples are derived by using the Rule for Conditional Actions. The key point is noticing that each of the conditions of the conditional actions involved refers to a web page different from the web page  $hp\text{age}(user)$  referred to in the precondition of the Hoare triple. The proof thus consists of using the fact that initially  $Bhp\text{age}(user)$  and the invariant  $inv$  to derive an inconsistency which immediately yield the desired Hoare triples by means of the Rule for Conditional Actions.

To prove the Hoare triple for  $B(hp\text{age}(user) \vee ContentCart) \wedge G(bought(book)) \rightarrow do(goto\_website(Am.com))$  we use the effect axiom (5) for  $goto\_website$  and the frame axiom (6):

$$\begin{aligned}
& \{Bhp\text{age}(user)\} \\
(5) \quad & goto\_website(Am.com) \\
& \{BAm.com\}
\end{aligned}$$

and

$$\begin{aligned}
& \{\neg Bin\_cart(book) \wedge \neg Bbought(book)\} \\
(6) \quad & goto\_website(Am.com) \\
& \{\neg Bin\_cart(book) \wedge \neg Bbought(book)\}
\end{aligned}$$

By using the axiom

$$\{Gbought(book)\} goto\_website(Am.com) \{Bbought(book) \vee Gbought(book)\}$$

the Conjunction Rule and the Rule for Conditional Actions it is then not difficult to obtain the desired conclusion.

## 6 Possible Extensions of GOAL

Although the basic features of the language GOAL are quite simple, the programming language GOAL is already quite powerful and can be used to program real agents. In particular, GOAL only allows the use of basic actions. There are, however, several strategies to deal with this restriction. First of all, if a GOAL agent is proven correct, *any* scheduling of the basic actions *that is weakly fair* can be used to execute the agent. More specifically, an interesting possibility is to define a mapping from GOAL agents to a particular agent architecture (cf. also [4]). As long as the agent architecture implements a weakly fair scheduling policy, concerns like the efficiency or flexibility may determine the specific mapping that is most useful with respect to available architectures.

A second strategy concerns the grain of atomicity that is required. If a coarse-grained atomicity of basic actions is feasible for an application, one might

consider taking complex plans as atomic actions and instantiate the basic actions in GOAL with these plans (however, termination of these complex plans should be guaranteed). Finally, in future research the extension of GOAL with a richer notion of action structure like for example plans could be explored. This would make the programming language more practical. The addition of such a richer notion, however, is not straightforward. At a minimum, more bookkeeping seems to be required to keep track of the goals that an agent already has chosen a plan for and which it is currently executing. This bookkeeping is needed, for example, to prevent the selection of more than one plan to achieve the same goal. Note that this problem was dealt with in GOAL by the immediate and complete execution of a selected action. It is therefore not yet clear how to give a semantics to a variant of GOAL extended with complex plans.

The ideal, however, would be to combine the language GOAL which includes declarative goals with our previous work on the agent programming language 3APL which includes planning features into a single new programming framework. Let us elaborate a little on the (im-)possibilities, here. A 3APL-Goal is a program or procedural goal, written P-Goal, and defined as either a basic action  $\text{Bact} \subseteq \text{Goal}$ , a test on the beliefs of the agent  $\varphi?$  or composed as a sequence  $(\pi_1; \pi_2)$  or a choice  $(\pi_1 + \pi_2)$ . However, also goal-variables  $X$  are allowed in P-Goals. Central in 3APL is the so-called *practical reasoning rule* of the form

$$\pi_h \leftarrow \varphi \mid \pi_b \in$$

which should be read as: ‘if  $\pi_h$ , the goal in the head of the rule, is the agent’s current (procedural) P-Goal, and he believes that  $\varphi$  is the case, then the rule allows the agent replace  $\pi_h$  with the goal in the body,  $\pi_b$ ’

Apart from introducing more complex action structures, it would also be particularly interesting to extend GOAL with high-level communication primitives. Because both declarative knowledge as well as declarative goals are present in GOAL, communication primitives could be defined in the spirit of speech act theory [29]. The semantics of, for example, a request primitive could then be formally defined in terms of the knowledge and goals of an agent. Moreover, such a semantics would have a computational interpretation because both beliefs and goals have a computational interpretation in our framework.

Finally, there are a number of interesting extensions and problems to be investigated in relation to the programming logic. For example, it would be interesting to develop a semantics for the programming logic for GOAL that would allow the nesting of the belief and goal operators. In the programming logic, we cannot yet nest knowledge modalities which would allow an agent to reason about its own knowledge or that of other agents. Moreover, it is not yet possible to combine the belief and goal modalities. It is therefore not possible for an agent to have a goal to obtain knowledge, nor can an agent have explicit rather than implicit knowledge about its own goals or those of other agents. So far, the use of the B and G operators in GOAL is, first of all, to distinguish between beliefs and goals. Secondly, it enables an agent to express that it does not have a particular belief or goal (consider the difference between

$\neg B\phi$  and  $B\neg\phi$ ). Another important research issue concerns an extension of the programming framework to incorporate first order languages and extend the programming logic with quantifiers. Finally, more work needs to be done to investigate and classify useful correctness properties of agents. In conclusion, whereas the main aim may be a unified programming framework which includes both declarative goals and planning features, there is still a lot of work to be done to explore and manage the complexities of the language GOAL itself.

## 7 Conclusion

Although a programming language dedicated to agent programming is not the only viable approach to building agents, we believe it is one of the more practical approaches for developing agents. Several other approaches to the design and implementation of agents have been proposed. One such approach promotes the use of *agent logics* for the specification of agent systems and aims at a further refinement of such specifications by means of an associated design methodology for the particular logic in use to implementations which meet this specification in, for example, an object-oriented programming language like Java. In this approach, there is no requirement on the existence of a natural mapping relating the end result of this development process - a Java implementation - and the formal specification in the logic. It is, however, not very clear how to implement these ideas for agent logics incorporating both informational and motivational attitudes and some researchers seem to have concluded from this that the notion of a motivational attitude (like a goal) is less useful than hoped for.

Still another approach consists in the construction of *agent architectures* which ‘implement’ the different mental concepts. Such an architecture provides a template which can be instantiated with the relevant beliefs, goals, etc. Although this second approach is more practical than the first one, our main problem with this approach is that the architectures proposed so far tend to be quite complex. As a consequence, it is quite difficult to understand what behaviour an architecture that is instantiated will generate.

For these reasons, our own research concerning intelligent agents has focused on the *programming language* 3APL which supports the construction of intelligent agents, and reflects in a natural way the intentional concepts used to design agents (in contrast with the approach discussed above which promotes the use of logic, but at the same time suggests that such an intermediate level is not required).

Nevertheless, in previous work the incorporation of declarative goals in agent programming frameworks has, to our knowledge, not been established. It has been our aim in this paper to show that it is feasible to incorporate declarative goals into a programming framework (and there is no need to dismiss the concept). Moreover, our semantics is a computational semantics and it is rather straightforward to implement the language, although this may require some restrictions on the logical reasoning involved on the part of GOAL agents.

Let us briefly indicate how incorporating declarative goals in the language

3APL might proceed. To this end, let us rename the goals in 3APL to *plans*  $\pi$ , which are either a basic action  $\mathbf{a}(t_1, \dots, t_n)$  on terms  $t_i$ , a test  $\varphi?$  or combined in sequential composition  $(\pi_1; \pi_2)$  or nondeterministic choice  $(\pi_1 + \pi_2)$ . One may also use *goal-variables*—  $X, Y, \dots$  in goals. Central in 3APL are the so-called *Practical Reasoning Rules*, in its most general form written as

$$\pi_h \leftarrow \varphi \mid \pi_b \in \text{Rule}$$

In this paper, we provided a complete programming theory. The theory includes a concrete proposal for a programming language and a formal, operational semantics for this language as well as a corresponding proof theory based on temporal logic. The logic enables reasoning about the dynamics of agents and about the beliefs and goals of the agent at any particular state during its execution. The semantics of the logic is provided by the GOAL program semantics which guarantees that properties proven in the logic are properties of a GOAL program. By providing such a formal relation between an agent programming language and an agent logic, we were able to bridge the gap between theory and practice. Moreover, a lot of work has already been done in providing practical verification tools for temporal proof theories [33].

Finally, our work shows that the (re)use of ideas and techniques from concurrent programming can be very fruitful. In particular, we have used many ideas from concurrent programming and temporal logics for programs in developing GOAL. It remains fruitful to explore and exploit ideas and techniques from these areas.

## References

- [1] P. Gärdenfors Alchourrón, C. E. and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.
- [2] Gregory R. Andrews. *Concurrent Programming: Principles and Practice*. The Benjamin/Cummings Publishing Company, 1991.
- [3] K.R. Apt and E-R. Olderog. *Verification of Sequential and Concurrent Programs*. Springer, New York, 1991.
- [4] K. Mani Chandy and Jayadev Misra. *Parallel Program Design*. Addison-Wesley, 1988.
- [5] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
- [6] Philip R. Cohen and Hector J. Levesque. Communicative Actions for Artificial Agents. In *Proceedings of the International Conference on Multi-Agent Systems*. AAAI Press, 1995.

- [7] E.W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Communications of the ACM*, 18:453–457, 1975.
- [8] R. Fagin and J. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence*, 34:39–76, 1988.
- [9] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge MA, 1994.
- [10] Giuseppe de Giacomo, Yves Lespérance, and Hector Levesque. ConGolog, a Concurrent Programming Language Based on the Situation Calculus. *Artificial Intelligence*, 121(1-2):109–169, 2000.
- [11] J.Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–101, 1987.
- [12] Koen V. Hindriks, Frank S. de Boer, Wiebe van der Hoek, and John-Jules Meyer. An Operational Semantics for the Single Agent Core of AGENT-0. Technical Report UU-CS-1999-30, Department of Computer Science, University Utrecht, 1999.
- [13] Koen V. Hindriks, Frank S. de Boer, Wiebe van der Hoek, and John-Jules Ch. Meyer. A Formal Embedding of AgentSpeak(L) in 3APL. In G. Antoniou and J. Slaney, editors, *Advanced Topics in Artificial Intelligence (LNAI 1502)*, pages 155–166. Springer-Verlag, 1998.
- [14] Koen V. Hindriks, Frank S. de Boer, Wiebe van der Hoek, and John-Jules Ch. Meyer. Formal Semantics for an Abstract Agent Programming Language. In M.P. Singh, A. Rao, and M.J. Wooldridge, editors, *Intelligent Agents IV (LNAI 1365)*, pages 215–229. Springer-Verlag, 1998.
- [15] Koen V. Hindriks, Frank S. de Boer, Wiebe van der Hoek, and John-Jules Ch. Meyer. Agent Programming in 3APL. *Autonomous Agents and Multi-Agent Systems*, 2(4):357–401, 1999.
- [16] Koen V. Hindriks, Yves Lespérance, and Hector J. Levesque. An Embedding of ConGolog in 3APL. Technical Report UU-CS-2000-13, Department of Computer Science, University Utrecht, 2000.
- [17] N.R. Jennings and M.J. Wooldridge. *Agent technology: Foundations, Applications, and Markets*. Springer, 1997.
- [18] W.L. Johnson, editor. *Proc. of the 1st Internatioanl Conference on Autonomous Agents*. ACM Press, 1997.
- [19] Bernd van Linder, Wiebe van der Hoek, and John-Jules Ch. Meyer. Formalising motivational attitudes of agents: On preferences, goals, and commitments. In M.J. Wooldridge, J.P. Müller, and M. Tambe, editors, *Intelligent agents II (LNAI 1037)*, pages 17–32. Springer-Verlag, 1996.
- [20] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.

- [21] J.-J.Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*. Number 41 in Cambridge Tracks in Theoretical Computer Science. Cambridge University Press, 1995.
- [22] John-Jules Ch. Meyer, Wiebe van der Hoek, and Bernd van Linder. A Logical Approach to the Dynamics of Commitments. *Artificial Intelligence*, 113:1–40, 1999.
- [23] G. Priest, R. Routley, and J. Norman, editors. *Paraconsistent Logic: Essays on the Inconsistent*. Philosophia Verlag, München, 1989.
- [24] Anand S. Rao. AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language. In W. van der Velde and J.W. Perram, editors, *Agents Breaking Away (LNAI 1038)*, pages 42–55. Springer-Verlag, 1996.
- [25] Anand S. Rao. Decision Procedures for Propositional Linear-Time Belief-Desire-Intention Logics. In M.J. Wooldridge, J.P. Müller, and M. Tambe, editors, *Intelligent Agents II (LNAI 1037)*, pages 33–48. Springer-Verlag, 1996.
- [26] Anand S. Rao and Michael P. Georgeff. Intentions and Rational Commitment. Technical Report 8, Australian Artificial Intelligence Institute, Melbourne, Australia, 1990.
- [27] S. Russell and P. Norvig. *Artificial Intelligence, A Modern Approach*. Prentice Hall, 1995.
- [28] V. Saraswat, M Ringard, and P Panangaden. Semantic foundations of concurrent constraint programming. In *Proceedings of the 18th. ACM Symposium on Principles of Programming Languages (POPL '91)*, pages 333–352, 1991.
- [29] John R. Searle. *Speech acts*. Cambridge University Press, 1969.
- [30] Yoav Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.
- [31] E. Thijsse. On total awareness logics (with special attention to monotonicity constraints and flexibility). In M. de Rijke, editor, *Diamonds and Defaults*, volume 229 of *Synthese Library*. Kluwer Academic Publishers, Dordrecht, 1993.
- [32] Sarah Rebecca Thomas. *PLACA, An Agent Oriented Programming Language*. PhD thesis, Department of Computer Science, Stanford University, 1993.
- [33] Tanja Vos. *UNITY in Diversity*. PhD thesis, Department of Computer Science, Utrecht University, 2000.

- [34] Wayne Wobcke. On the Correctness of PRS Agent Programs. In N.R. Jennings and Y. Lespérance, editor, *Intelligent Agents VI (LNAI 1757)*. Springer-Verlag, 2000.
- [35] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.